Eduskunnan tietojärjestelmien integraatiotestauksen automatisointi

Nina Katila

Helsinki 18.04.2020

HELSINGIN YLIOPISTO
Tietojenkäsittelytieteen osasto

HELSINGIN YLIOPISTO – HELSINGFORS UNIVERSITET – UNIVERSITY OF HELSINKI

Tiedekunta – Fakultet – Faculty	Koulutusohjelma - Studieprogram - Study Programme
Matemaattis-luonnontieteellinen tiedekunta	Tietojenkäsittelytieteen maisteriohjelma

Tekijä – Författare – Author

Nina Katila

Työn nimi – Arbetets titel – Title

Eduskunnan tietojärjestelmien integraatiotestauksen automatisointi

Ohjaajat - Handledare – Supervisors

Antti-Pekka Tuovinen ja Eila Nummi

Työn laji – Arbetets art – Level	Aika – Datum – Month and year	Sivumäärä – Sidoantal – Number of pages
Pro gradu -tutkielma	18.4.2020	65 sivua + 1 liite

Tiivistelmä – Referat – Abstract

Tutkielmassa tarkastellaan tieteellisen tutkimuksen ja ammattikirjallisuuden pohjalta keinoja ja näkökohtia tietojärjestelmien integraatiotestauksen automatisointiin. Tutkimusmetodologiana on tapaustutkimus (Case Study). Tutkimuksen tapausympäristönä on eduskunnan lainsäädäntötyön tietojärjestelmien välisen integraatiotestauksen automatisoinnin edellytykset ja eri toteutusvaihtoehdot. Tutkielmaa varten tietoa on kerätty eduskunnan tietojärjestelmien dokumentaatiosta sekä eduskunnan tietojärjestelmien asiantuntijoilta. Eduskunnan integraatiotestauksen työnkulut ja testauksen haasteet perustuvat havaintoihin, joita on tehty osallistumalla eduskunnan integraatiotestaukseen noin vuoden ajan. Automatisointivaihtoehtojen analysointi ja evaluointi perustuvat jatkuvaan yhteistyöhön eduskunnan lainsäädäntötyön tietojärjestelmien ja integraatiojärjestelmän asiantuntijoiden kanssa.

Eduskunnan lainsäädäntötyön tietojärjestelmät ovat toiminnallisesti ja hallinnollisesti itsenäisiä sekä toteuttajiltaan, toteutukseltaan ja iältään erilaisia. Koska itsenäisten järjestelmien ohjelmakoodi ei ole järjestelmien välillä saatavilla, on integraatiotestauksen automatisoinnin ratkaisun perustuttava järjestelmien käyttöliittymien kautta saavutettavaan koodiin. Tutkimuksessa havaittiin, että ohjelmistorobotiikan (Robotic Process Automation, RPA) avulla voidaan jäljitellä eduskunnan testaajien järjestelmien käyttöliittymien kautta suorittamaa integraatiotestausta. Tutkimuksessa havaittiin, että testauksen automatisointiin ja ohjelmistorobotiikkaan soveltuvan automatisointikehyksen avulla on mahdollista automatisoida eduskunnan tietojärjestelmien integraatiotestaus. Ohjelmistorobotiikalla integraatiotestit saadaan suoritettua manuaalista testausta merkittävästi nopeammin ja vähemmillä resursseilla.

Käyttöliittymiin perustuvan testausautomaation merkittävin haittapuoli on testien ylläpidon kustannukset. Modulaarisen avainsanapohjaisen automatisointikehyksen avulla voidaan tietojärjestelmien automatisoituja testejä ja niiden osia käyttää uudelleen integraatiotestauksen automatisoinnissa ja näin säästää kustannuksissa.

ACM Computing Classification System (CCS):
Software and its engineering - Software creation and management - Software verification and Validation

Avainsanat – Nyckelord – Keywords

Integraatiotestaus, automaattinen testaus, toiminnallinen testaus, järjestelmien testaus

Säilytyspaikka – Förvaringställe – Where deposited

Muita tietoja – Övriga uppgifter – Additional information

Sisältö

1 Johdanto

Organisaatioiden toimintaprosessit vaativat usein monen eri tietojärjestelmän käyttöä ja palveluiden välittämistä näiden järjestelmien välillä. Käytössä olevat järjestelmät saattavat olla hyvin eri ikäisiä ja toteutukseltaan erilaisia. Alallaan kauan toimineilla organisaatioilla voi olla käytössään hyvinkin vanhoja järjestelmiä (Legacy Systems), joiden kanssa uusien järjestelmien tulee integroitua tarjotakseen palveluita toisilleen ja toimiakseen yhteen toimintaprosessien edellyttämällä tavalla. Toiminnallisesti ja hallinnollisesti itsenäisten erilaisten järjestelmien integraatiot muodostavat uuden järjestelmän, järjestelmien järjestelmän (System of Systems), joka suorittaa laajempaa tehtävää kuin mitä järjestelmät itsenäisinä voivat suorittaa (Neves, Bertolino, De Angelis & Garcés, 2018, s.29).

Järjestelmien järjestelmän integraatiotestauksella varmistetaan, että sen itsenäiset erilliset osajärjestelmät toimivat yhteen oikealla tavalla osajärjestelmille määriteltyjen vaatimusten mukaisesti (Neves, Bertolino, De Angelis & Garcés, 2018, s. 31). Järjestelmien järjestelmässä erilliset järjestelmät toimivat yhteistyössä suorittaakseen laajempaa tehtävää kuin mitä järjestelmät itsenäisinä voivat suorittaa ja testauksessa verifioidaan tämän toteutuminen testaamalla järjestelmien järjestelmä yhtenä kokonaisuutena (Neves, Bertolino, De Angelis & Garcés, 2018, s. 29 ja s. 31). Järjestelmien järjestelmän integraatiotestaus käsittää kaikkien erillisten osajärjestelmien testauksen niiltä osin, kun nämä integraatioidensa kautta pyytävät ja tarjoavat palvelua toisilleen.

Järjestelmien järjestelmän manuaalinen integraatiotestaus vaatii sen jokaisen erillisen järjestelmän testausta integraatioihin osallistuvien käyttötapausten osalta. Manuaalinen testaus sitoo henkilöstöresursseja, vie paljon aikaa ja on virhealtista (Dustin, Rashka, & Paul, 1999, s. 18 ja s. 63). Järjestelmien järjestelmän integraatiotestauksen automatisoinnilla pyritään lisäämään testauksen tehokkuutta ja parantamaan osajärjestelmien välisten integraatioiden laadunvarmistusta. Testauksen automatisoinnin tavoitteiden saavuttaminen tehokkuuden parantamiseksi edellyttää testauksen automatisoinnin suunnittelua (Jussi Kasurinen, Ossi Taipale, & Kari Smolander, 2010, s. 15).

Tässä tutkielmassa tarkastellaan tieteellisen tutkimuksen ja ammattikirjallisuuden pohjalta keinoja ja näkökohtia järjestelmien järjestelmän integraatiotestauksen automatisointiin sekä arvioidaan näiden soveltuvuutta eduskunnan lainsäädäntötyön tietojärjestelmien välisten integraatioiden testauksen automatisointiin. Tutkielman tavoitteena on löytää eduskunnan tietojärjestelmien integraatiotestauksen automatisointiin sopivat vaihtoehdot, joiden pohjalta integraatiotestauksen automatisointi voidaan toteuttaa. Tutkimusmetodologiana on tapaustutkimus (Case Study), jossa on tutkittu eduskunnan lainsäädäntötyön tietojärjestelmien välisen integraatiotestauksen automatisoinnin edellytyksiä ja eri toteutusvaihtoehtoja. Tutkimuksen tuloksena on

analysoitu ja evaluoitu eduskunnan integraatiotestauksen kehitysmahdollisuuksia ja automatisoinnin vaihtoehtoja.

Tutkielmaa varten tietoa on kerätty eduskunnan tietojärjestelmien dokumentaatiosta sekä eduskunnan tietojärjestelmien asiantuntijoilta. Eduskunnan integraatiotestauksen työnkulut, testauksen vaatima testaajien määrä, testaukseen kuluva aika ja testauksen haasteet perustuvat havaintoihin, joita on tehty osallistumalla eduskunnan integraatiotestaukseen noin vuoden ajan. Automatisointivaihtoehtojen analysointi ja evaluointi perustuvat jatkuvaan yhteistyöhön eduskunnan lainsäädäntötyön tietojärjestelmien ja integraatiojärjestelmän asiantuntijoiden kanssa.

Eduskunnan lainsäädäntötyön tietojärjestelmät ovat toiminnallisesti ja hallinnollisesti itsenäisiä sekä toteuttajiltaan, toteutukseltaan ja iältään erilaisia. Manuaalinen eduskunnan järjestelmien integraatiotestaus vaatii jokaisen erillisen järjestelmän testaajien testaustyötä tietyssä lainsäädäntötyöprosessin mukaisessa järjestyksessä, mikä puolestaan vaatii testauksen tarkkaa suunnittelua ja koordinointia. Riittävän kattavaa integraatiotestausta ja integraatiotestauksen automatisointia varten on tunnistettava tärkeimmät ja erillisten järjestelmien integraatioiden osalta kattavimmat eduskunnan lainsäädäntötyöprosessin avainskenaariot.

Lasilaatikkotestauksen (White-box testing) tekniikoita ei voida käyttää integraatiotestauksen automatisoinnissa, sillä itsenäisten järjestelmien ohjelmakoodi ei ole muiden erillisten järjestelmien nähtävissä tai saatavilla. Järjestelmien välisen integraatiotestauksen automatisoinnin ratkaisun on perustuttava järjestelmien käyttöliittymien kautta saavutettavaan koodiin ja toiminnallisuuteen. Automatisoitujen integraatiotestien tarkoituksena ei ole testata erillisten järjestelmien toimintaa laajemmin kuin integraatioihin osallistuvien käyttötapausten osalta. Erillisten järjestelmien integraatioihin osallistuvien käyttötapausten edellytyksenä tai esiehtona olevien käyttötapausten suorittaminen tulee automatisoida järjestelmien oman testauksen automatisoinnin yhteydessä.

Eduskunnan integraatiotestauksen automatisointivaihtoehtojen käytännön kokeilussa integraatiotestauksen automatisointia toteutettiin testauksen automatisointiin ja ohjelmistorobotiikkaan soveltuvan Robot Framework automatisointikehyksen avulla. Ohjelmistorobotiikan (Robotic Process Automation, RPA) avulla voidaan jäljitellä eduskunnan testaajien järjestelmien käyttöliittymien kautta suorittamaa integraatiotestausta. Testauksen automatisointiin ja ohjelmistorobotiikkaan soveltuvan automatisointikehyksen avulla on mahdollista automatisoida eduskunnan tietojärjestelmien integraatiotestaus. Ohjelmistorobotiikalla integraatiotestit saadaan suoritettua manuaalista testausta merkittävästi nopeammin ja vähemmillä resursseilla. Robot Framework automatisointikehyksen avulla toteutetut eduskunnan integraatiotestit todentavat vaatimusten mukaisen vuorovaikutuksen järjestelmien välillä, sillä robotti testaa avainskenaarioihin sisältyvät integraatiot käyttöliittymien kautta täsmälleen kuten eduskunnan tietojärjestel-

mien pääkäyttäjät ne testaavat avainskenaarioiden sisältämien integraatiotestitapausten testauksessa.

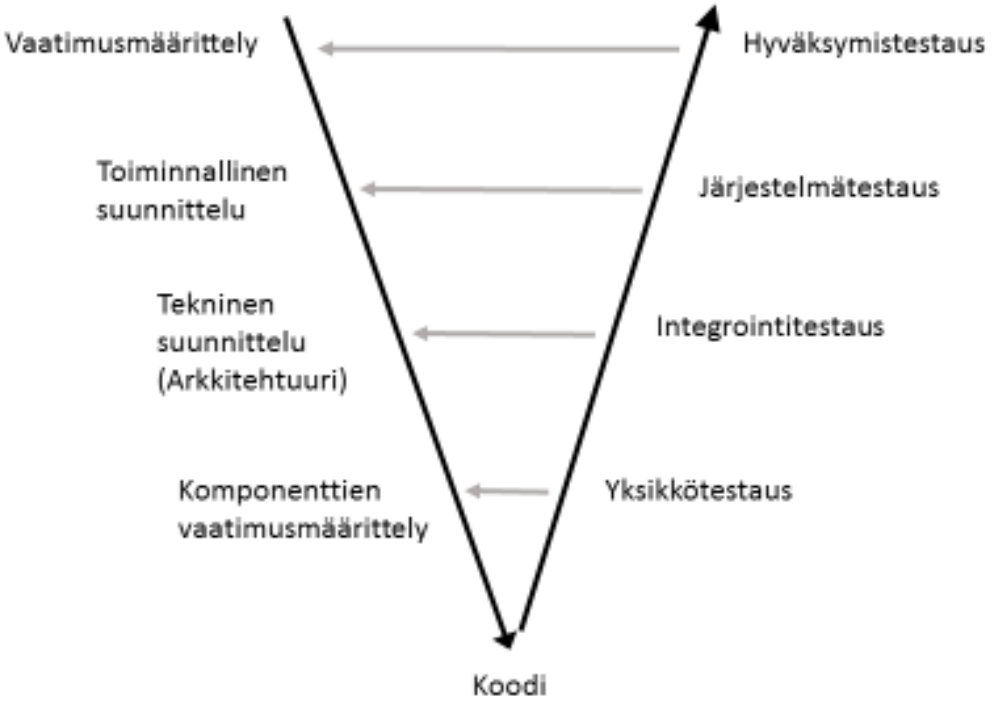
Käyttöliittymiin perustuvan testausautomaation merkittävin haittapuoli on testien ylläpitoon liittyvät kustannukset. Käyttöliittymiin perustuvien testien ylläpidon kustannuksiin voidaan vaikuttaa testiskriptien modulaarisella arkkitehtuurilla, joka mahdollistaa testiskriptin osien itsenäisen ylläpidon erillään skriptin muista osista (Alégroth, Feldt & Kolström, 2016, s.76). Modulaarisen avainsanapohjaisen automatisointikehyksen avulla voidaan automatisoituja integraatiotestejä ja niiden osia käyttää uudelleen. Käytettäessä samaa automatisointikehystä järjestelmien välisen integraatiotestauksen ja erillisten järjestelmien oman testauksen automatisoinnissa voidaan kustannuksissa säästää. Erillisten järjestelmien automatisoituja testejä voidaan käyttää uudelleen integraatiotestauksen automatisoinnissa. Testien osien ylläpito tapahtuu järjestelmien testien ylläpidon yhteydessä, minkä jälkeen järjestelmien välisten integraatiotestien ylläpito kohdistuu vain jo ylläpidettyjen testien osien uudelleen käyttöön.

Tutkielman rakenne on seuraava. Tutkielman luvussa 2 esitellään tieteellisistä tutkimuksista ja ammattikirjallisuudesta kerätty näkemys järjestelmän integraatiotestauksesta, järjestelmien järjestelmän integraatiotestauksesta ja integraatiotestauksen automatisoinnista. Luvussa 3 esitetään tutkimuskohteena olevan eduskunnan lainsäädäntötyö tähän liittyvine tietojärjestelmineen. Luku 4 käsittää kuvauksen eduskunnan lainsäädäntötyön tietojärjestelmien välisistä integraatiosta ja näiden testauksesta soveltaen eduskunnan toimintaympäristöön luvussa 2 mainittuja malleja. Luvussa 5 esitetään eduskunnan järjestelmien integraatiotestauksessa esiin nousseet haasteet sekä toimintaympäristön vaikutus integraatiotestaukseen. Luvussa 6 kuvataan eduskunnan integraatiotestauksen automatisoinnin suunnitelma ja luvussa 7 evaluoidaan automatisoinnin suunnitelman eri toteutusvaihtoehtoja. Tutkielman yhteenveto on luvussa 8.

2 Integraatiotestaus ja sen automatisointi

Testaus on tärkein osa ohjelmistokehitystä, ja riittävä tarkoituksenmukainen testaus on ehdoton edellytys onnistuneelle ohjelmistokehitykselle ja ohjelmiston käyttöönotolle (Gandhi, 2016, s. 3867). Testauksen tehtäviä suoritetaan koko ohjelmistokehityksen elinkaaren ajan. Onnistuneen testauksen edellytyksiä ovat testauksen jakautuminen tasoihin ja testauksen alkaminen yksikkötestaustasolta edeten seuraavaksi integraatiotestaustasolle ja sen jälkeen järjestelmätestaustasolle (Gandhi, 2016, s. 3867).

Testauksen jakautumista eri testaustasoihin ja testaustasojen liittymistä ohjelmistokehityksen vaiheisiin kuvataan usein erilaisin mallein riippuen tarkastelun kohteeksi valitusta ohjelmistokehitysmallista. Tunnetuin testauksen malli on V-malli, jossa testaus jaetaan ohjelmistokehitysvaiheisiin liittyviin testaustasoihin. V-mallin testauksen tasot ovat yksikkötestaus, integrointitestaus, järjestelmätestaus ja hyväksymistestaus (Spillner, Linz, &Schaefer, 2014, s.69-70). Kuvassa 1 on esitetty testauksen V-malli.



Kuva 1. Testauksen V-malli (Spillner, Linz, &Schaefer, 2014, s.69)

V-mallin vasen haara kuvaa ohjelmistokehityksen vaiheet ja oikea haara kuvaa näitä vaiheita vastaavat testaustasot (Spillner, Linz, &Schaefer, 2014, s.69-70). Jokaisen testaustason testauksessa käytetään testauksen pohjamateriaalina testaustasoa vastaavan ohjelmistokehityksen vaiheen

tuotosta eli määrittelyjä ja kuvauksia. Integrointitestauksessa testataan, että ohjelmiston komponentit toimivat yhteen teknisen suunnitteluvaiheen määrittelyiden mukaisesti (Spillner, Linz, &Schaefer, 2014, s.69-70).

Ohjelmistokehityksessä integroinnilla tarkoitetaan erillisten osien, kuten yksittäisten komponenttien tai yksittäisten järjestelmien yhdistämistä yhdeksi kokonaisuudeksi. V-mallin integrointitestaustasolla ohjelmistokehittäjät muodostavat yksikkötestaustasolla tai järjestelmätestaustasolla testatuista osista kokonaisuuksia, ja tästä syystä on tärkeää, että erilliset osat on ensin riittävän perusteellisesti testattu yksinään. Yksikkötestauksen jälkeen alkavan integrointitestauksen aloituksen edellytyksenä on, että ohjelmiston yksittäiset osat on ensin testattu yksikkötestein ja yksikkötestauksessa havaitut virheet on korjattu (Spillner, Linz, &Schaefer, 2014, s.81).

Integrointitestauksen pohjamateriaalina on teknisen suunnittelun yhteydessä tuotetut ohjelmiston ja järjestelmän suunnitelmat, arkkitehtuuri, työnkulut ja käyttötapaukset (Spillner, Linz, &Schaefer, 2014, s.82 ja Müller, Beer, Klonk, & Verma, 2010, s. 24). Integrointitestauksessa testataan osien toiminta yhdessä. Testaajat keskittyvät testaamaan osien välistä kommunikaatiota eikä jo yksikkötestauksessa testattua yksittäisen osan toiminnallisuutta (Müller, Beer, Klonk, & Verma, 2010, s. 24). Integrointitestauksen kohteena ovat osien väliset rajapinnat ja liittymät, ja sen tehtävänä on löytää yhteistoiminnan väliset ongelmat ja eristää niiden aiheuttajat (Spillner, Linz, &Schaefer, 2014, s.84).

2.1 Integraatiotestaus järjestelmien järjestelmässä

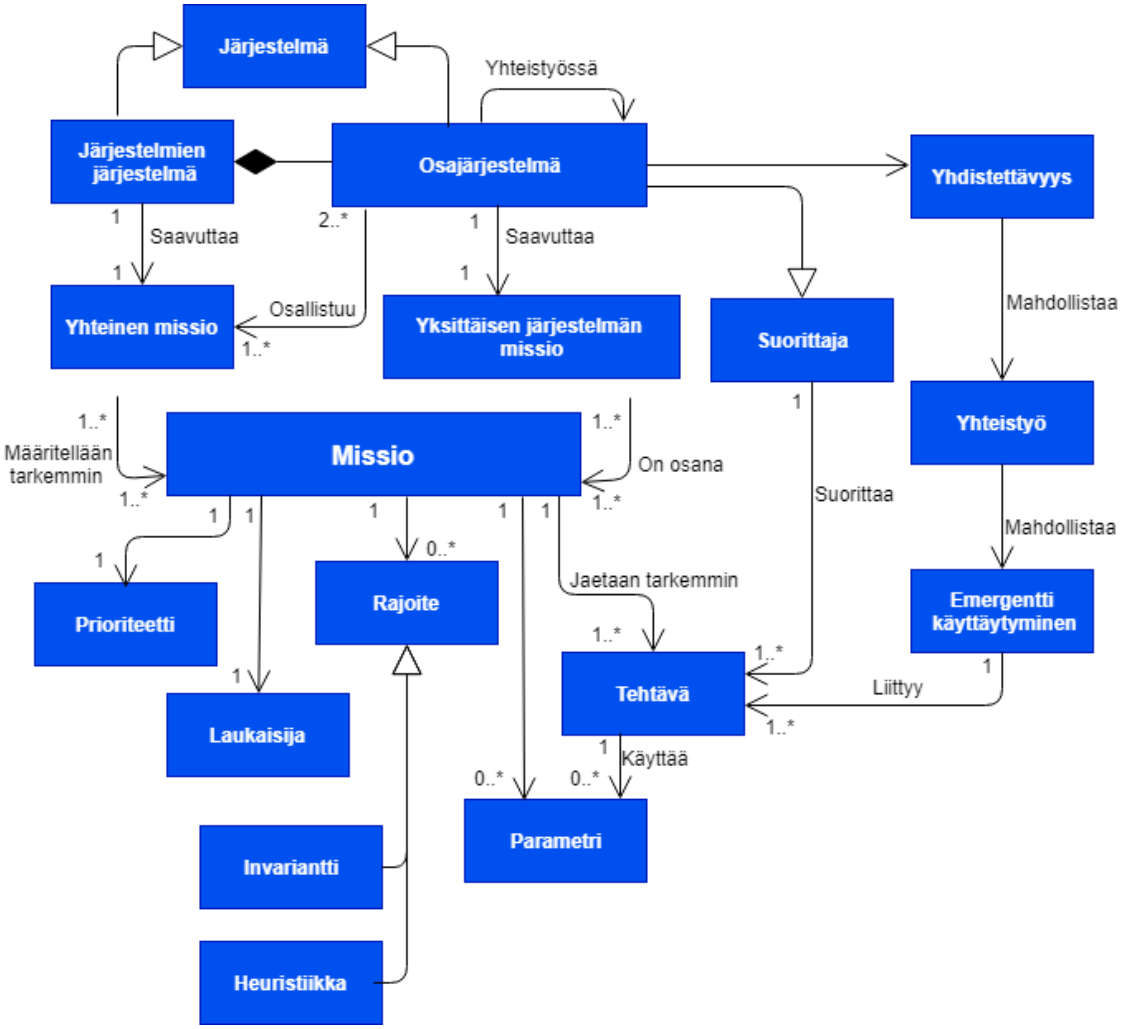
Järjestelmien järjestelmää (System of systems, SoS) kuvataan toiminnallisesti ja hallinnollisesti itsenäisten erilaisten järjestelmien integraatioiden muodostamana uutena järjestelmänä, jossa siihen liittyvien erilaisten järjestelmien tulisi toimia yhteen (Ali, Petersen & Mäntylä, 2012, s.211). Järjestelmien järjestelmässä erilliset järjestelmät toimivat yhteistyössä suorittaakseen laajempaa tehtävää kuin mitä järjestelmät itsenäisinä voivat suorittaa (Neves, Bertolino, De Angelis & Garcés, 2018, s.29). Järjestelmät integroituvat yhteen järjestelmien järjestelmään vain joidenkin käyttötapaustensa osalta, ja nämä käyttötapaukset ovat tärkeitä ymmärtää tehtäessä suunnitteluratkaisuja järjestelmien järjestelmää koskien (Kazman, Nielsen & Schmid, 2013, s. 141).

Kehittämisen vaiheiden näkökulmasta järjestelmien järjestelmä on kuin mikä tahansa järjestelmä, joskin monimutkaisempi (Clark, 2009, s. 385). Myös järjestelmien järjestelmän kehittämisen vaiheet sekä testaustasot noudattavat V-mallia (Clark, 2009, s. 384). Järjestelmien järjestelmän yksikkötestaus käsittää yksittäisten järjestelmien testaukset kaikkine vaiheineen kyseisten järjestelmien toimittajien toimesta (Neves, Bertolino, De Angelis & Garcés, 2018, s. 31). Järjestelmien järjestelmän integraatiotestauksella varmistetaan, että sen osat eli osajärjestelmät

toimivat yhteen oikealla tavalla osajärjestelmille määritettyjen vaatimusten mukaisesti (Neves, Bertolino, De Angelis & Garcés, 2018, s. 31). Osajärjestelmiin kohdistuu vaatimuksia järjestelmien järjestelmän koko kehitysprosessia ohjaavan toiminnallisen tavoitteen, mission, kautta (Silva, Batista & Oquendo, 2015, s. 346). Järjestelmien järjestelmän testauksessa verifioidaan tämän mission toteutuminen testaamalla järjestelmien järjestelmä yhtenä kokonaisuutena (Neves, Bertolino, De Angelis & Garcés, 2018, s. 31). Järjestelmien järjestelmän testausta voikin harkita lähestyttävän missiolähtöisesti (mission-driven testing), jolloin testaus perustuu järjestelmien järjestelmällä saavutettavaan missioon (Neves, Bertolino, De Angelis & Garcés, 2018, s. 31). Järjestelmien järjestelmän suunnittelussa tärkeä haaste on mission ja siihen liittyvän tiedon systemaattinen mallintaminen (Silva, Batista & Oquendo, 2015, s. 346). Mission systemaattisesta mallintamisesta voidaan hyötyä missiolähtöisessä testauksessa. Kuvassa 2 on esitetty järjestelmien järjestelmän mission käsitemalli, jossa kuvataan eri käsitteiden liittyminen pääkäsitteeseen eli missioon (Silva, Batista & Oquendo, 2015, s. 347).

Järjestelmien järjestelmän mission käsitemallissa missiot muodostuvat osajärjestelmien omista missioista sekä järjestelmien järjestelmän yhteisestä missiosta. Määrittämällä missiolle prioriteetti, laukaiseva tekijä, rajoitteet, parametrit sekä tarkemmat osatehtävät, joihin suorittajat ja osajärjestelmät yhteistyön kautta liittyvät, saadaan missio systemaattisesti mallinnettua. Missiota ja sen mallintamista voidaan hyödyntää järjestelmien järjestelmän missiolähtöisessä integraatiotestauksessa.

Järjestelmien järjestelmän kehittämisen ja testaamisen on tunnistettu olevan erittäin haastavaa johtuen mm. järjestelmien toiminnallisesta ja hallinnollisesta riippumattomuudesta toisistaan ja järjestelmien erilaisesta integroitumisesta järjestelmien järjestelmään (Ali, Petersen & Mäntylä, 2012, s.211). Eri järjestelmien kehityssyklien ja elinkaarien yhteensovittaminen tuovat omat haasteensa järjestelmien järjestelmän kehittämiseen ja testaamiseen (Dahmann, Lane, Rebovich & Lowry, 2010, s. 2). Vaikka osajärjestelmät ovat toiminnallisesti riippumattomia toisistaan, saattaa osajärjestelmien integraatioidensa kautta tarjoamat palvelut tai tiedot järjestelmien järjestelmälle riippua toisistaan. Esimerkiksi usean sairaalan potilastietohallintajärjestelmien muodostama järjestelmien järjestelmä voi tarjota kaikille siihen osallistuville sairaaloille täydelliset tiedot potilaan hoidosta näissä sairaaloissa vain siinä tapauksessa, että kaikkien näiden sairaaloiden potilastietohallintajärjestelmät lähettävät tietoa järjestelmien järjestelmälle (Lewis, Morris, Place, Simanta, Smith & Wrage, 2008, s. 3). Mikäli tällaisia voimakkaita riippuvuuksia on olemassa, täytyy järjestelmien järjestelmän elinkaaren aikaiset kehityssyklit synkronoida näiden osajärjestelmien elinkaaren kehityssykleihin (Lewis, Morris, Place, Simanta, Smith & Wrage, 2008, s. 3). Järjestelmien järjestelmän jokaisen kehityssyklin testauksessa on syytä kiinnittää huomiota testitapausten valintaan ja priorisointiin sekä erityisesti siihen, milloin testaus on syytä suorittaa (Neves, Bertolino, De Angelis & Garcés, 2018, s. 31).



Kuva 2: Järjestelmien järjestelmän mission käsitemalli (Silva, Batista & Oquendo, 2015, s. 347)

Järjestelmien järjestelmän ja sen osajärjestelmien toiminnallisesta ja hallinnollisesta itsenäisyydestä johtuvia haasteita on esitetty taulukossa 1. Taulukossa on esitetty omilla riveillään myös järjestelmien järjestelmän emergentin käyttäytymisen (Emergent behaviour) sekä osajärjestelmien maantieteellisestä hajaantumisesta aiheutuvia haasteita. Emergentillä käyttäytymisellä tarkoitetaan järjestelmien järjestelmän kokonaisuutena saavuttamia toivottuja tai ei toivottuja lopputuloksia, joita ei yksittäisten järjestelmien ole yksinään toimiessaan mahdollista saavuttaa (Neves, Bertolino, De Angelis & Garcés, 2018, s. 30).

Ominaisuus	Seuraus	Testauksen haasteet	Testausvaihe	Hyödynnettävä lähestymistapa
Toiminnallinen itsenäisyys	Riippumattomuus osajärjestelmien välillä Osajärjestelmät voivat toteuttaa useita toimintoja, mutta vain muutamaa käytetään järjestelmien järjestelmän mission täyttämiseen	Mitä tulisi testata? Mitkä ovat oikeat sopivat kriteerit osajärjestelmälle? Mitkä osajärjestelmän tarjoamat testitapaukset tulisi valita?	Yksikkötestaus	Komponentti-pohjaiset järjestelmät
Hallinnollinen itsenäisyys	Osajärjestelmien elinkaarten riippumattomuus toisistaan Yhteentoimivuuden varmistaminen Yksittäisen osajärjestelmän saavutettavuudeen varmistaminen Luotettavuuden varmista-minen	Kuka vastaa validoinnista ja verifioinnista? Miten osajärjestelmä hyväksytään? Miten varmistetaan oikean tiedon välittyminen oikealle järjestelmälle? Kuinka organisoida testaus?	Integraatio-testaus	Palvelukeskei-nen arkkiteh-tuuri Komponentti-pohjaiset jär-jestelmät
Emergentti käyttäytyminen	Dynaaminen uudelleenkonfigurointi Sopimaton käyttäytyminen Missio voi muuttua ajon aikana tilanteesta ja kontekstista riippuen	Kuinka havaita ei toivottu toiminta ja tuhota se nopeasti ja ketterästi? Kuinka varmistaa, että osajärjestelmän käyttäytyminen on linjassa mission kanssa? Kuinka varmistaa, että on tehty oikeanlainen muutos?	Järjestelmien järjestelmä	Adaptiivinen testaus
Maantieteel-linen hajaantu-minen	Hajauttaminen Rinnakkaisuus	Kuinka validointi ja verifiointi tulisi organisoida?	Järjestelmien järjestelmä	Rinnakkainen testaus

Taulukko 1: Yhteenveto järjestelmien järjestelmän ominaisuuksista ja lähestymistavoista muista yhteyksistä testaukseen liittyen (Neves, Bertolino, De Angelis & Garcés, 2018, s. 32).

Taulukon Testausvaihe –sarake sisältää ne testausvaiheet, joissa samalla rivillä mainitut testauksen haasteet tulee käsitellä. Hyödynnettävä lähestymistapa -sarake tarjoaa mahdollisia muissa yhteyksissä esille tulleita lähestymistapoja tähän käsittelyyn (Neves, Bertolino, De Angelis & Garcés, 2018, s. 31). Esimerkiksi toiminnallisesta itsenäisyydestä seuraavat testauksen haasteet ovat tuttuja myös suurten komponenttipohjaisten järjestelmien testauksesta, joissa komponentteja voivat toteuttaa eri toimittajat (Neves, Bertolino, De Angelis & Garcés, 2018, s. 31). Tällöin lasilaatikkotestauksen (White-box testing) tekniikoita ei voida käyttää, sillä komponenttien lähdekoodit eivät ole tiedossa tai nähtävillä (Neves, Bertolino, De Angelis & Garcés, 2018, s. 31). Kun tarkastellaan osajärjestelmien välisiä yhteyksiä verkkona, niin lasilaatikkotestauksen tekniі-

koista polkukattavuutta voidaan kuitenkin hyödyntää integraatiotestauksen testijoukon suunnittelussa ja näin saada testijoukko sisältämään kaikki ne testitapaukset, joissa edes kerran käydään jokainen yksittäinen polku eli osajärjestelmien välinen yhteys läpi (Zapata, Akundi, Pineda & Smith, 2013, s. 256).

Koska järjestelmät voivat liittyä ja poistua järjestelmien järjestelmästä, integraatioihin liittyen voi kohdistua uudelleenkonfigurointitarpeita muissa osajärjestelmissä (Neves, Bertolino, De Angelis & Garcés, 2018, s. 30). Järjestelmien itsenäisyyden on havaittu myös johtavan puutteisiin yhteisten prosessien noudattamisessa, puutteellisiin vastuisiin eri testaustasoilla, puutteelliseen ohjeistukseen yhteisen testauksen ja yhteisten testaustyökalujen osalta sekä puutteelliseen testitapausten valintaan (Ali, Petersen & Mäntylä, 2012, s.216). Järjestelmien järjestelmän testauksen haasteista nousee esille erityisesti testiympäristö, oikeiden testitapausten löytäminen, riittävän ja tarkoituksenmukaisen testauksen tason määrittäminen, järjestelmien järjestelmästä johtuvan virheen määrittäminen sekä vastuullisen testaajan osoittaminen (Neves, Bertolino, De Angelis & Garcés, 2018, s. 31).

Järjestelmien järjestelmälle asetettujen vaatimusten kääntämisessä testattaviksi vaatimuksiksi tarvitaan laajaa järjestelmien rajojen ylittävää toimialan asiantuntijuutta (Ali, Petersen & Mäntylä, 2012, s.216). Testitapausten määrän parempi kontrolli, relevanttien testitapausten valitseminen, turhien ja vanhentuneiden testitapausten poistaminen sekä testijoukon hallinta nimetyn vastuuhenkilön toimesta nähdään tärkeinä kehityskohteina järjestelmien järjestelmän testauksessa (Ali, Petersen & Mäntylä, 2012, s. 217-218). Vakioituja käyttöskenaarioita kuvaavat konkreettiset käyttötapaukset, joissa eri järjestelmät toimivat yhdessä, ovat testauksessa hyödyllisempiä, kuin paneutuminen siihen, mikä voi mennä väärin (Ali, Petersen & Mäntylä, 2012, s.219). Järjestelmien järjestelmän testauksessa on tärkeää johtaa integraatiotestit niistä avainskenaarioista, joilla varmistetaan vuorovaikutus ympäristön kanssa ja osajärjestelmien välillä (Lima, 2018, s. 956). Avainskenaarioiden visualisoinnilla voidaan löytää järjestelmien järjestelmän integraatiotestauksen automatisoinnin kohteet.

2.2 Integraatiotestauksen automatisointi

Testauksen automatisoinnilla tarkoitetaan testausaktiviteettien suorittamista ja hallintaa testityökaluohjelmistoja hyödyntämällä (Dustin, Rashka, & Paul, 1999, s. 27). Testityökaluohjelmisto tukee yhtä tai useampaa testausaktiviteettia kuten suunnittelua, testien määrittelyä, testiaineistojen hallintaa, testauksen suorittamista ja testien analysointia (International Software Testing Qualifications Board [ISTQB] testaussanasto, 2017). Testauksen automatisoinnilla pyritään lisäämään testauksen tehokkuutta. Manuaalinen testaus sitoo henkilöstöresursseja, vie paljon aikaa ja on virhealtista (Dustin, Rashka, & Paul, 1999, s. 18 ja s. 63). Testauksen automatisoinnin

tavoitteita ovat testaukseen kuluvien resurssien vähentäminen ja testauksen kattavuuden parantaminen (Dustin, Rashka, & Paul, 1999, s. 16).

Testauksen automatisoinnin merkittävin hyöty mahdollisten kustannussäästöjen lisäksi on testikattavuuden ja suuremman testimäärän kautta saavutettava laadun paraneminen (Karhu, Repo, Taipale & Smolander, 2009, s. 206). Automatisoinnin merkittävimmäksi haitaksi nähdään kustannukset, joita aiheutuu automatisoinnin käyttöönotosta, ylläpidosta ja koulutuksesta (Karhu, Repo, Taipale & Smolander, 2009, s. 207). Testauksen automatisointi ei välttämättä vähennä validointiin ja verifiointiin käytettyä aikaa, mutta se vähentää tuntuvasti kehitykseen kaikkiaan käytettyä aikaa tarjoamalla nopeamman jajatkuvan palautteen järjestelmän laadusta (Alégroth, Feldt & Kolström, 2016, s.78). Näin ollen testauksen automatisoinnin ensisijainen hyöty tuleekin parantuneen laadun eikä testauksen kustannusten vähenemisestä (Alégroth, Feldt & Kolström, 2016, s.78).

Järjestelmän testattavuuteen ja testauksen automatisoinnin kannattavuuteen vaikuttaa järjestelmälle suunniteltu tekninen arkkitehtuuri ja sen testaus. Yksittäisen järjestelmän testattavan arkkitehtuurin ehtoja ovat ohjelmiston kerrostaminen, ja testauksen automatisointi integraatiotestaustasolla testipetiin kuuluvia ei vielä olemassa olevien komponenttien tynkiä (stub) ja jäljitelmiä (mocks) käyttäen (Linz, 2014, s.96). Testattavuus integraatiotestaustasolla on heikkoa, mikäli ei ole riittäviä mahdollisuuksia jäljitellä keskeneräisiä järjestelmän osia, jotta voidaan suorittaa eristettyjä testejä rajapinnoille ja systemaattisia testejä virheenhallinnalle (Berner, Weber, &Keller, 2005, s.576). Järjestelmien järjestelmän integraatiotestauksessa ja sen automatisoinnissa on mahdollista käyttää osajärjestelmien palveluihin kohdistuvien pyyntöjen ja vastausten jäljitelmiä tai kokonaisten palveluiden jäljitelmiä, nk. virtuaalisia palveluja. Jäljitelmien laatimiseen on olemassa valmiita työkaluja, kuten SoapUI (Software testing help, 7 Best Service Virtualization Tools in 2019). Palveluita voi virtualisoida analysoimalla osajärjestelmien rajapintakuvauksia (kuten WSDL), nauhoittamalla transaktioita tai määrittelemällä rajapinnan käyttäytyminen manuaalisesti (Eniser, Sen & Polat, 2018, s. 1576). Palveluiden välisten transaktioiden nauhoittamisessa ja jäljentämisessä voidaan hyödyntää koneoppimista. Järjestelmien järjestelmän integraatiotestauksessa jäljitelmät ovat tarpeen esimerkiksi silloin, kun osajärjestelmä on poissa päältä tai uusi integroitava osajärjestelmä on vielä kehityksen alla.

Järjestelmätestauksen automatisointi pohjautuu järjestelmätestauksen manuaalisiin testitapauksiin, joita ei voi suorittaa tehokkaalla tavalla toistuvasti ilman niiden muuntamista ensin suoritettaviksi testiskripteiksi (Thummalapenta, Sinha, Singhania, & Chandra, 2012, s. 890). Järjestelmätestiskriptien luonnissa voidaan käyttää apuna erilliseen taulukkoon tallennettuja avainsanoja, joita käytetään testiskriptin sisältämissä ohjelmakutsuissa ja testin lopputilan tarkistuksessa (Thummalapenta, Sinha, Singhania, & Chandra, 2012, s. 881). Toinen vaihtoehto

avainsanaohjatun testauksen lisäksi on hyödyntää nauhoitus- / toistotyökalua. Käyttäjän toimintaa voidaan nauhoittaa käyttöliittymän kautta, kun käyttäjä suorittaa manuaalista testiä, ja muodostaa tästä myöhemmin toistettava testiskripti (Thummalapenta, Sinha, Singhania, & Chandra, 2012, s. 881).

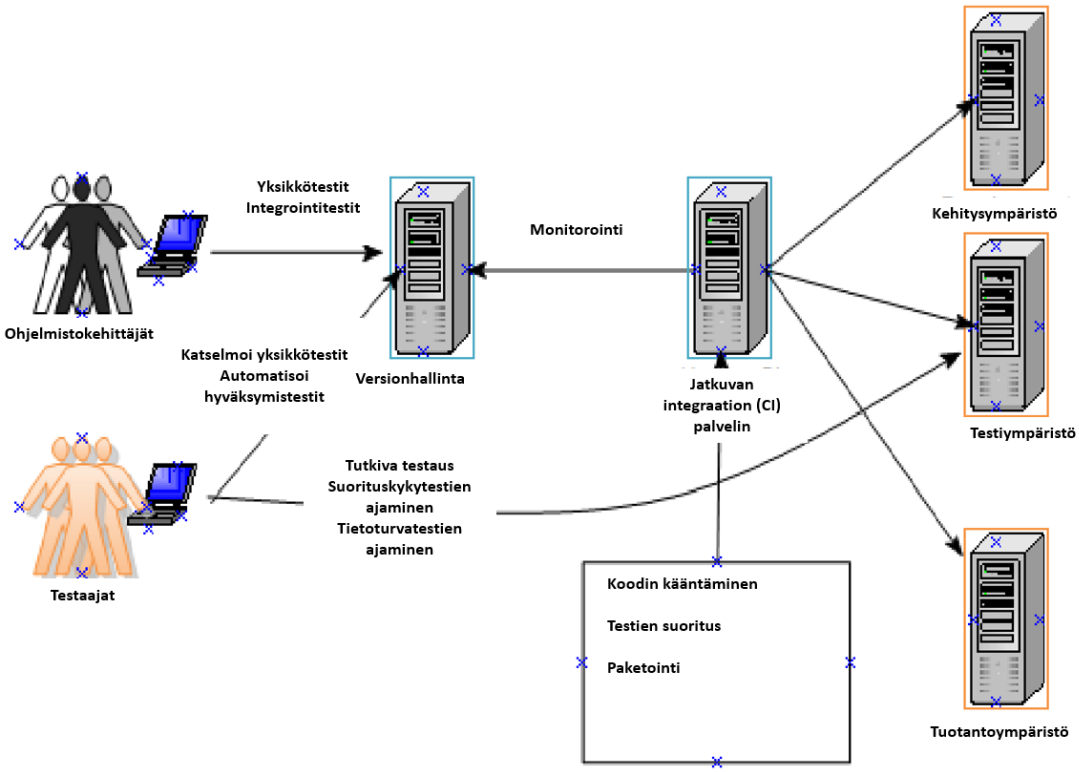
Järjestelmien järjestelmän manuaalisessa integraatiotestauksessa testataan sen kaikkien osajärjestelmien ne testitapaukset, joissa osajärjestelmät integroituvat muihin osajärjestelmiin. Järjestelmien järjestelmän testaus vaatii useampien itsenäisten toteutuksiltaan ja toteuttajiltaan erilaisten järjestelmien toimintojen suorittamista, eikä avainsanoihin pohjautuva testiskriptien laatiminen ohjelmakutsuihin ja testin lopputilan tarkistukseen yli osajärjestelmien ole mahdollista ilman kaikkien näiden osajärjestelmien käyttöliittymäkoodin hyvää tuntemusta. Käyttöliittymän kautta nauhoitetut testiskriptit ovat puolestaan herkkiä rikkoutumaan pienimmästäkin käyttöliittymää koskevasta muutoksesta (Thummalapenta, Sinha, Singhania, & Chandra, 2012, s. 881). Visuaalisen käyttöliittymän kautta testaukseen liittyy useita haasteita kuten testiskriptien pitkä suoritusaika, korkeat ylläpitokustannukset ja korkeat virheellisten positiivisten (false positive) virhehavaintojen analysointiin kuluvat kustannukset (Alégroth, Karlsson & Radway, 2018, s. 180-181).

Testausautomaatiotyökalujen käyttö tehokkaampaan testitapausten suunnitteluun ja testauksen raportointiin säästää testauksen kustannuksia helposti enemmän kuin testauksien suorituksen automatisointi (Berner, Weber, &Keller, 2005, s.573). Testauksen hallinnan työkalun avulla vaatimukset saadaan tuotua testitapauksien suunnitteluun ja jokaisen vaatimuksen testauksen tilannetta voidaan seurata ja jäljittää (Spillner, Linz, & Schaefer, 2014, s. 286). Tunnettuja testauksen hallinnan työkaluja ovat mm. Zephyr, TestFLO for JIRA ja Xray (Software testing help, List Of The Best Test Management Tools In 2020).

Yleisimpiä testiympäristössä koettuja testauksen automatisoinnin ongelmia ovat manuaaliset asennukset, manuaaliset konfiguraatioiden muutokset, pääsyn puute olennaiseen infrastruktuuriin kuten konfiguraatioiden hallintajärjestelmään ja testaajalle näkymätön testausautomaatio (Berner, Weber, &Keller, 2005, s.577). Näiden ongelmien ehkäisemiseksi integrointi- ja järjestelmätestaus kannattaa heti ohjelmistokehityksen alussa automatisoida käyttämällä jatkuvaa integraatiota (Continuous Integration, CI). Kuvassa 3 on havainnollistettu testauksen automatisointia jatkuvan integraation avulla.

Ohjelmistokoodi lähetetään ohjelmistokehittäjien työasemilta versionhallintapalvelimelle, josta jatkuvan integraation palvelin sen havaitsee ja noutaa. Jatkuvan integraation avulla uudet toiminnallisuudet integroidaan jo olemassa olevaan koodiin useita kertoja päivässä, ja integroinnin yhteydessä kaikki sillä hetkellä olemassa olevat automaattiset yksikkö-, integraatio-, järjestelmä- ja hyväksymistestit ajetaan automaattisesti. Tämä nopeuttaa huomattavasti testausta ja

mahdollistaa riittävän kattavan testauksen lyhyidenkin ohjelmistokehitysjaksojen aikana. Jatkuva integraatio pakottaa testien ajamisen päivittäin, mikä auttaa pitämään automatisoidut testit ajan tasalla ja näin vähentää riskiä testien hajoamisesta käyttämättömyyden vuoksi (Berner, Weber, &Keller, 2005, s.577). Automatisoidut testit ovat avainasemassa jatkuvassa integraatiossa mahdollistaen pikaisen palautteen testauksen kohteen laadusta (Alégroth, Karlsson & Radway, 2018, s. 172)



Kuva 3. Automatisointi jatkuvassa integraatiossa (Collins, & de Lucena, 2012, s. 61).

3 Eduskunnan lainsäädäntötyön tietojärjestelmät

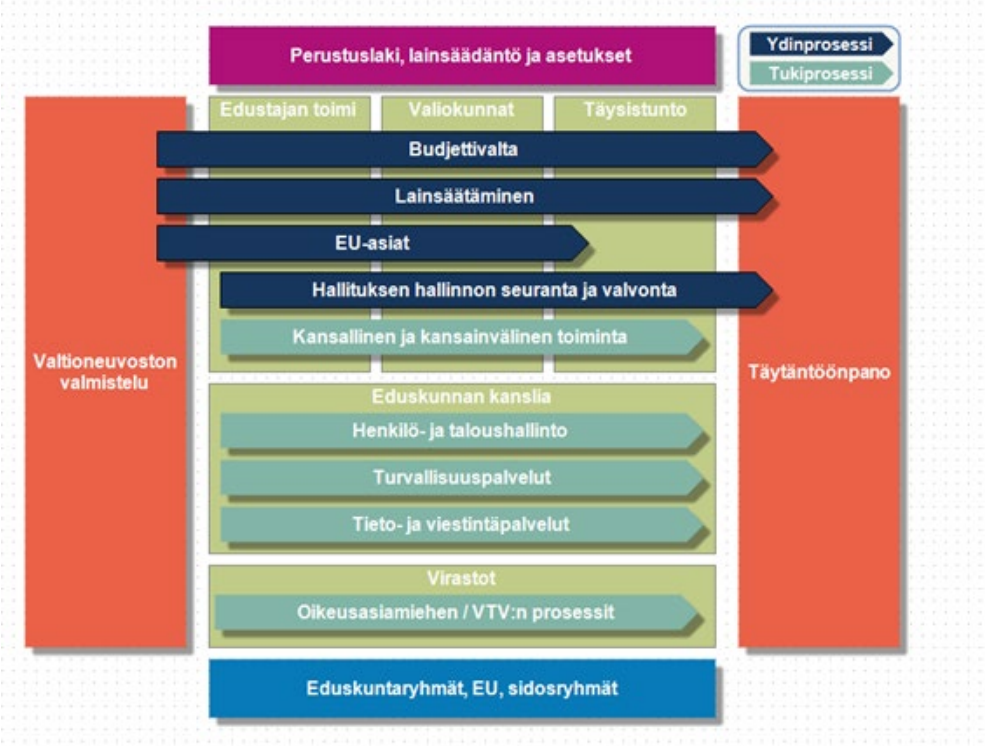
Tämän tutkielman tarkoituksena on tarkastella eduskunnan lainsäädäntötyön tietojärjestelmien välisten integraatioiden testauksen automatisoinnin edellytyksiä ja toteutusvaihtoehtoja. Eduskunnan lainsäädäntötyön tietojärjestelmien väliset integraatiot on toteutettu palveluväylänä (Enterprise Service Bus, ESB) toimivan integraatiojärjestelmän avulla. Palveluväylä mahdollistaa palvelukeskeisen arkkitehtuurin (Service Oriented Architecture, SOA) toimimalla välittäjänä väliohjelmistossa (Middleware), jonka kautta eri palvelut ovat saavutettavissa (Shenoy, Mohapatra & Swamy, 2013, s.1). Jokainen eduskunnan lainsäädäntötyön tietojärjestelmä on toiminnallisesti ja hallinnollisesti itsenäinen järjestelmä ja palvelee muita järjestelmiä ainoastaan palveluväylänä toimivan integraatiojärjestelmän kautta. Eduskunnan ICT-ympäristö on monitoimittajaympäristö, jossa eri toimittajien toimittamat tietojärjestelmät toimivat yhdessä eduskunnan lainsäädäntöprosessin eri vaiheissa. Seuraavissa tämän kappaleen alaluvuissa esitetään lyhyesti eduskunnan lainsäädäntöprosessi ja siihen liittyvät tietojärjestelmät.

3.1 Eduskunta

Keskeisimmät eduskunnan tehtävät ovat lakien säätäminen, valtion talousarviosta eli budjetista päättäminen, hallituksen toiminnan valvominen ja kansainvälisten velvoitteiden hyväksyminen (Suomen perustuslaki 11.6.1999/731, §3, §83, §90, §93, §94 ja §96). Eduskunta valitsee pääministerin, oikeusasiamiehen ja valtuutettuja valvomaan kansaneläkelaitoksen ja Suomen pankin toimintaa (Suomen perustuslaki 11.6.1999/731, §36, §38 ja §61). Eduskunta myös vaikuttaa Euroopan unionin asioihin.

Eduskunnan keskeisin ydinprosessi on lainsäätäminen, jossa keskeisin ja näkyvin toimija on täysistunto. Täysistunnossa päätetään muun muassa lakiesityksien käsittelystä ja siitä, mihin valiokuntaan lakiesitykset lähetetään. Muita ydinprosesseja ovat budjetista päättäminen eli budjettivalta, EU-asioiden käsittely ja hallituksen hallinnon seuranta ja valvonta. Kuvassa 4 on esitetty eduskunnan ydinprosessit ja niitä tukevat tukiprosessit.

Eduskunnan ydinprosesseissaan käsittelemät valtiopäiväasiat valmistellaan valtioneuvostossa. Yleisin valtioneuvoston valmisteleva ja eduskuntaan lähettämä valtiopäiväasia on hallituksen esitys, jossa useimmiten esitetään muutosta voimassa olevaan lakiin. Myös esitykset valtion talousarvioksi tai lisätalousarvioksi ovat hallituksen esitys –tyyppisiä valtiopäiväasioita. Hallituksen esityksiä ja kansanedustajien laatimia lakialoitteita nimitetään lakiesityksiksi. Lakiesityksen tyypillisin käsittelyprosessi eduskunnassa esitetään tarkemmin seuraavassa kappaleessa 3.2.



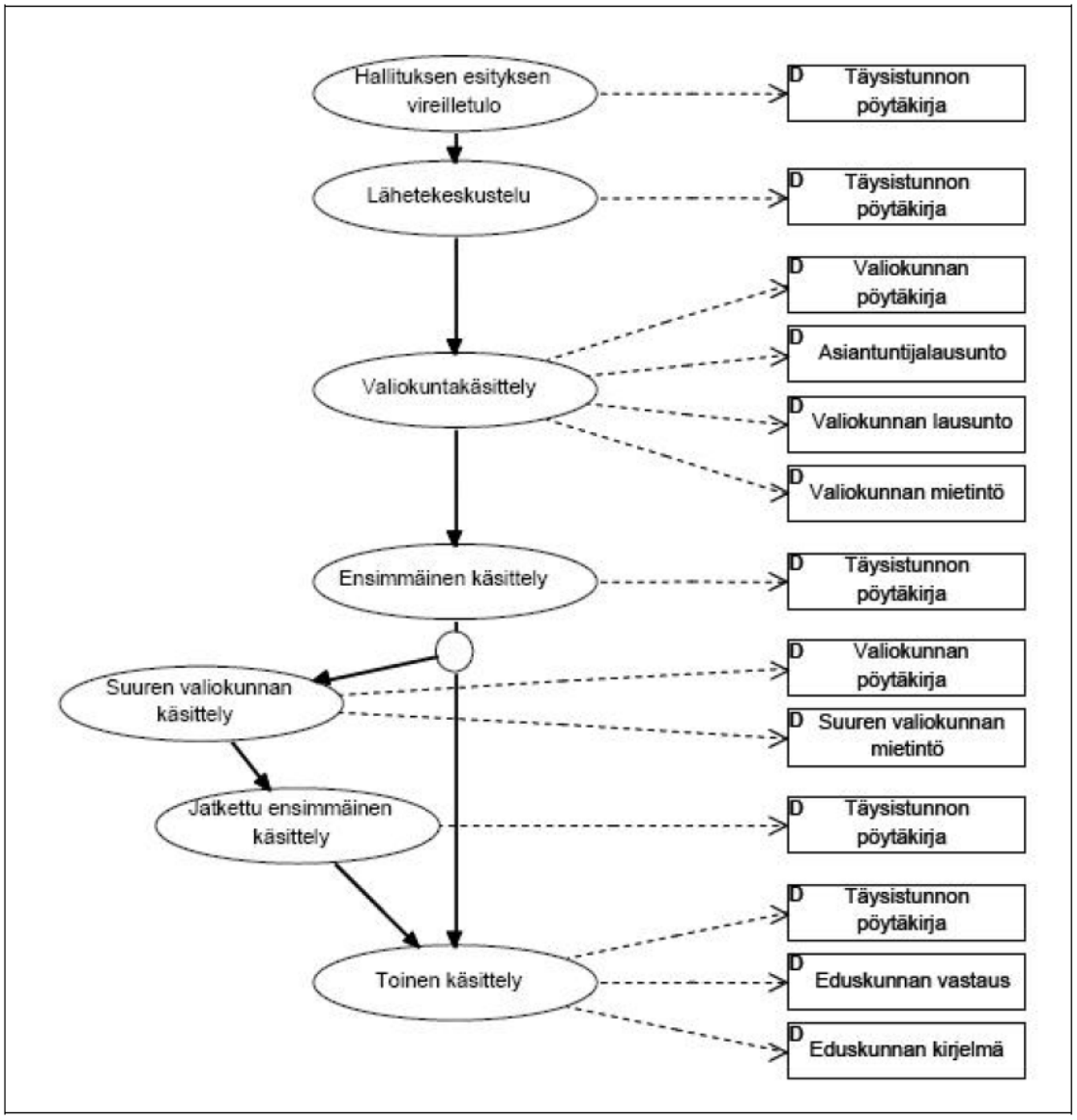
Kuva 4: Eduskunnan prosessikartta (Eduskunnan intranet, 2019)

3.2 Lainsäädäntötyö eduskunnassa

Eduskunnan keskeisin tehtävä on lain säätäminen. "Lain säätäminen tulee eduskunnassa vireille hallituksen esityksellä taikka kansanedustajan lakialoitteella, joka voidaan tehdä eduskunnan ollessa koolla" (Suomen perustuslaki 11.6.1999/731, §70).

Kuvassa 5 on esitetty lakiesityksen tyypillinen eduskuntakäsittely ja siinä syntyvä dokumentaatio (Nurmeksela, 2006, s. 49).

Lakiesityksen tyypillinen eduskuntakäsittely käsittää viisi täysistuntokäsittelyvaihetta: lakiesityksen ilmoittaminen saapuneeksi, lähetekeskustelu, mietinnön pöydällepano (ei ole kuvattu kuvassa), ensimmäinen käsittely ja toinen käsittely. Lakiesityksen tyypillinen käsittelyprosessi käsittää myös yhden valiokuntakäsittelyvaiheen, joka sisältää monivaiheisen valiokuntakäsittelyprosessin mietinnön laativassa valiokunnassa sekä lausunnon laativassa yhdessä tai useammassa valiokunnassa. Kuvassa on lisäksi kuvattu sivupolkuna mahdollinen lakiesityksen käsittely suuressa valiokunnassa.



Kuva 5. Lakiesityksen tyypillinen eduskuntakäsittely (Nurmeksela, 2006, s. 49).

3.3 Lainsäädäntötyön tietojärjestelmät eduskunnassa

Lainsäädäntötyön tietojärjestelmät tukevat eduskunnan prosessikartassa kuvassa 4 kuvattuja ydinprosesseja. Lainsäädäntötyön ydinprosesseihin liittyviä eduskunnan tietojärjestelmiä kutsutaan eduskunnassa ydinjärjestelmiksi. Eduskunnan ydinjärjestelmät ovat: edustaja- ja toimielintietojen hallintajärjestelmä, täysistunnon hallintajärjestelmä, täysistunnon äänitallenteiden hallintajärjestelmä, päättyneiden täysistuntojen hallintajärjestelmä, valtiopäiväasioiden ja -asiakirjojen hallintajärjestelmä ja julkaisujen hallintajärjestelmä.

Edustaja- ja toimielintietojen hallintajärjestelmällä käsitellään edustajiin ja toimielimiin, kuten

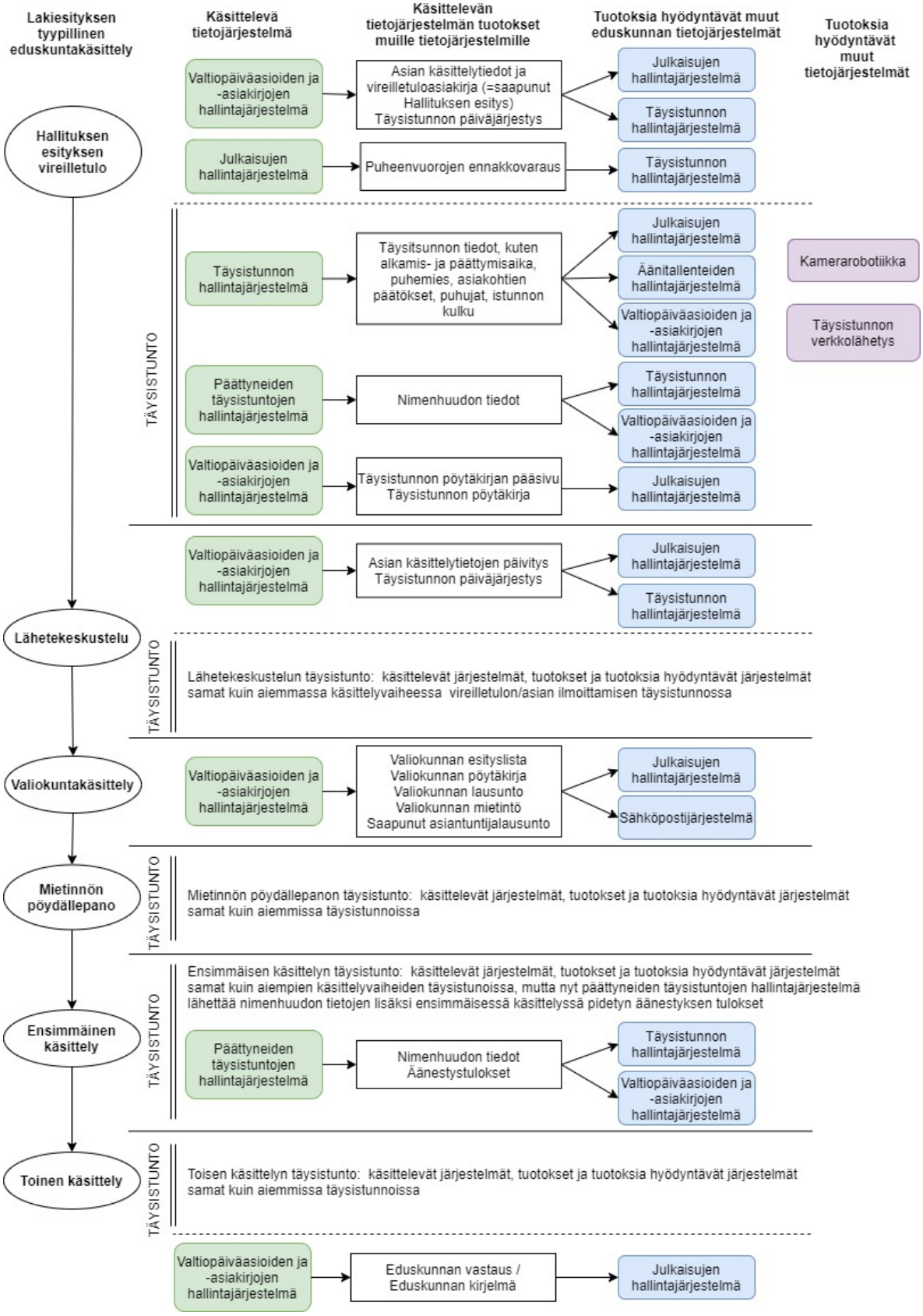
valiokuntiin, liittyviä tietoja. Lisäksi järjestelmässä pidetään yllä valtiopäivävuosiin, vaalipiireihin ja eduskuntaryhmiin liittyvää voimassaolevaa tietoa sekä kaikkien edellä mainittujen tietojen historiatietoja. Kaikki eduskunnan muut ydinjärjestelmät hyödyntävät toiminnassaan edustaja- ja toimielintietojen hallintajärjestelmän tietoja, joten järjestelmän merkitys tietolähteenä muille ydinjärjestelmille on suuri. Edustaja- ja toimielintietojen hallintajärjestelmä ei käytä prosesseissaan tietoja eduskunnan muista ydinjärjestelmistä.

Eduskunnan keskeisimmät tehtävät eli lakien säätäminen ja valtion talousarviosta päättäminen tapahtuvat täysistunnossa. Täysistunnon hallintajärjestelmä, äänitallenteiden hallintajärjestelmä ja päättyneiden täysistuntojen hallintajärjestelmä ovat täysistuntotyössä käytettäviä ydinjärjestelmiä, joilla hallitaan täysistuntotyössä syntyviä tietoja, kuten edustajien puheenvuorotiedot, edustajien äänestystiedot ja eduskunnan päätökset lakiesityksiin. Puheenvuorot ja päätöstiedot kirjataan täysistuntotyössä valtiopäiväasioiden ja –asiakirjojen hallintajärjestelmällä laadittaviin pöytäkirjoihin.

Valtiopäiväasioiden ja –asiakirjojen hallintajärjestelmällä hallitaan valtiopäiväasioiden eduskuntakäsittely sekä siihen liittyvät asiakirjat. Valtiopäiväasialla tarkoitetaan eduskuntakäsittelyyn saapuneita ja avattavia asioita kuten hallituksen esitys, lakialoite ja kansalaisaloite. Valtiopäiväasiakirjalla tarkoitetaan valtiopäiväasioiden käsittelyyn liittyviä ja käsittelyssä syntyviä asiakirjoja, kuten vireilletuloasiakirja, täysistunnon pöytäkirja, valiokunnan mietintö ja eduskunnan vastaus. Erilaisia valtiopäiväasiatyyppejä on noin 30, erilaisia valtiopäiväasiakirjatyyppejä on yli 100 ja asiankäsittelyprosesseihin liittyviä käsittelyvaiheita on noin 250. Luvussa 3.1 esitetty lakiesityksen tyypillinen käsittelyprosessi on yksi asiankäsittelyprosesseista. Valtiopäiväasian käsittelyprosessi riippuu valtiopäiväasian tyypistä ja eri käsittelyvaiheissa eduskunnan tekemistä päätöksistä. Jotkin valtiopäiväasiat eivät mene ollenkaan esimerkiksi valiokuntakäsittelyyn, kuten toimenpidealoitteet (TPA –asiat).

Julkaisujen hallintajärjestelmä käsittää eduskunnan sisäisen ja ulkoisen verkkopalvelun, jota mm. Avoin data –palvelu hyödyntää. Julkaisujen hallintajärjestelmä julkaisee muista ydinjärjestelmistä saamiaan tietoja sekä järjestelmään itseensä syötettyjä tietoja. Julkaisujen hallintajärjestelmässä eduskunnan sisäisen verkkopalvelun kautta edustajat voivat mm. varata puheenvuoroja seuraavan täysistunnon asiakohtiin täysistunnon ennakkovarausaikana.

Kuva 6 on laadittu luvussa 3.1 esitetyn lakiesityksen tyypillistä eduskuntakäsittelyä kuvaavan kuvan pohjalta ottaen mukaan käsittelyprosessiin liittyvät eduskunnan ydinjärjestelmät ja niiden olennaiset tuotokset eli palvelut muille tietojärjestelmille. Ydinjärjestelmät osallistuvat yhdessä laajempaan tehtävään, lakiesityksen eduskuntakäsittelyyn, tarjoamalla palveluitaan integraatioiden kautta muille ydinjärjestelmille ja näin täyttävät oman osansa yhteisestä missiosta.



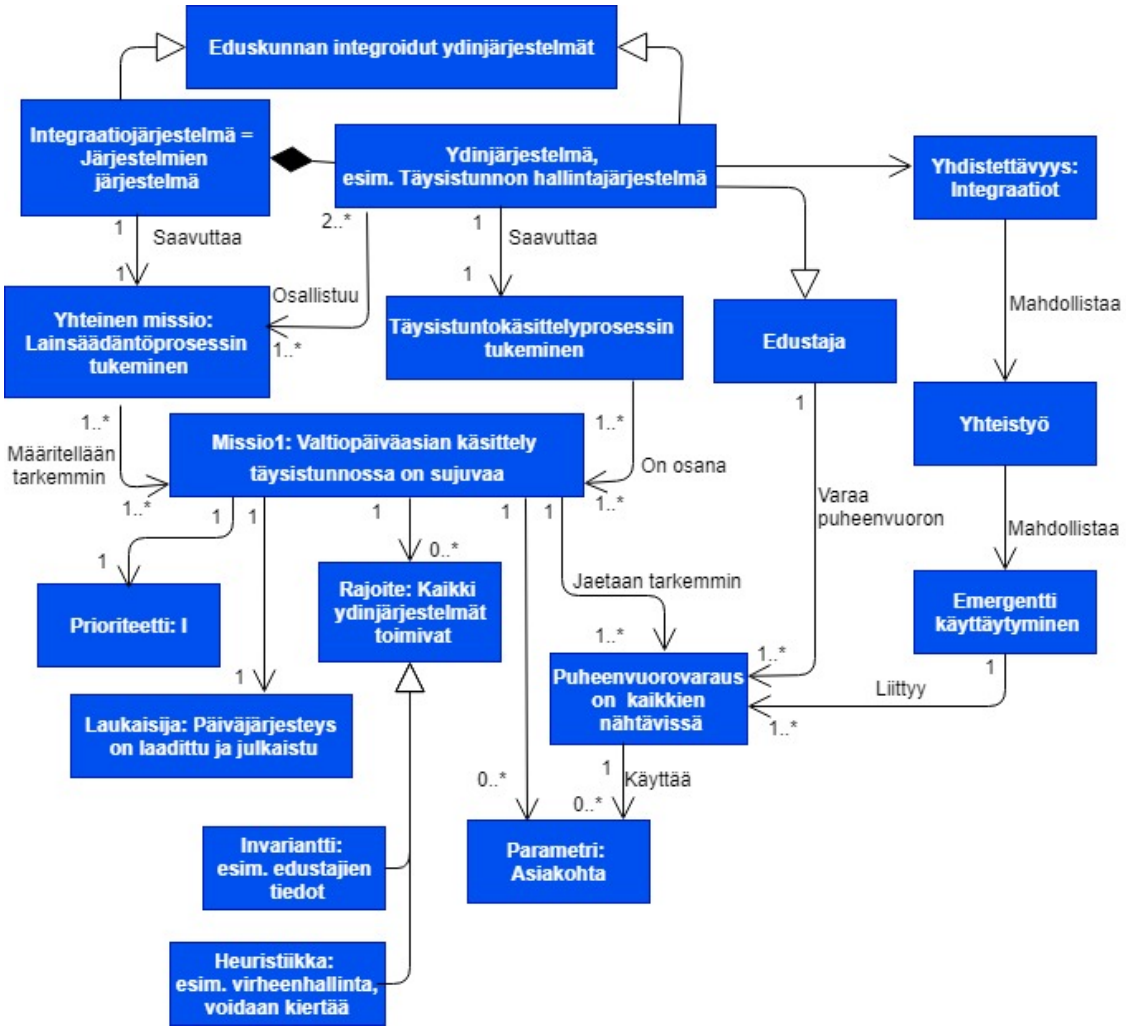
Kuva 6. Lakiesityksen tyypilliseen eduskuntakäsittelyyn liittyvät tietojärjestelmät.

Kuvalla 6 on pyritty havainnollistamaan ydinjärjestelmien integraatioidensa kautta tarjoamia palveluja sekä integraatioiden merkitys ja riippuvuus lakiesityksen eduskuntakäsittelyprosessin eri vaiheista.

Ydinjärjestelmät käsittelevät lainsäädäntötyön eri vaiheissa tuotettavaa tietoa ja lähettävät käsittelemäänsä tietoa muille järjestelmille. Ydinjärjestelmissä tuotettavien tietojen lähettäminen ja vastaanottaminen tapahtuvat järjestelmien välisten liittymien eli integraatioiden kautta. Järjestelmät eivät lähetä tai vastaanota tietoa toisiltaan suoraan, vaan erillisen integraatiojärjestelmän ja sinne toteutettujen järjestelmien välisten integraatioiden kautta.

4 Eduskunnan tietojärjestelmien väliset integraatiot ja niiden testaus

Eduskunnan tietojärjestelmien väliset integraatiot mahdollistavat eduskunnan tehtävien suorittamisen eri tietojärjestelmissä sekä eri tietojärjestelmien välisen yhteistyön. Järjestelmien välisellä integraatiotestauksella varmistetaan, että toisiinsa integroituneet tietojärjestelmät toimivat yhteen oikealla tavalla tietojärjestelmille määritettyjen vaatimusten mukaisesti. Testauksen tarkoitus on myös varmistaa tietojärjestelmien yhdessä suorittaman laajemman tehtävän ja mission toteutuminen. Kuvassa 7 on kuvattu eduskunnan järjestelmien järjestelmän missio kuvassa 2 sivulla 6 esitetyn mission käsitemallin pohjalta ottaen ydinjärjestelmistä esimerkiksi täysistunnon hallintajärjestelmä ja siitä esimerkiksi edustajan puheenvuorovaraus. Edustaja tekee puheenvuorovarauksen täysistunnon hallintajärjestelmässä ja varaus tulee näkyviin eduskunnan Täysistunto-verkkosivulla.



Kuva 7. Edustajan puheenvuorovaraus eduskunnan järjestelmien järjestelmän missiomallissa.

Eduskunnan järjestelmien järjestelmän missio on lainsäädäntöprosessin tukeminen. Missio on hyvin laaja käsittäen koko eduskunnan lainsäädäntöprosessin päästä päähän ja siksi se on tarpeen määrittää tarkemmin jakamalla missio osiin valtiopäiväasian käsittelyvaiheiden ja ydinjärjestelmien omien missioiden mukaisesti. Kuvan esimerkissä on määritetty missio tarkemmin täysistunnon hallintajärjestelmän ja yhden sen tehtävän, puheenvuoron varaaminen, avulla. Valtiopäiväasian käsittely täysistunnossa, kuten asiasta keskusteleminen ja äänestäminen, on oltava sujuvaa. Tämä edellyttää mm. edustajien mahdollisuutta varata puheenvuoroja asiakohtaan, puhemiehen mahdollisuutta jakaa puheenvuoroja ja kaikkien mahdollisuutta nähdä puheenvuorovaraukset sekä itse puheenvuorot. Jotta puheenvuorovaraus saadaan kaikkien nähtäville eduskunnan verkkosivuille, tarvitsee täysistunnon hallintajärjestelmän tarjota tämä tieto integraatioidensa kautta mm. julkaisujen hallintajärjestelmälle.

Järjestelmien järjestelmän testauksessa verifioidaan mission toteutuminen testaamalla järjestelmien järjestelmä yhtenä kokonaisuutena (Neves, Bertolino, De Angelis & Garcés, 2018, s. 31). Koska eduskunnan eri tietojärjestelmät ovat hallinnollisesti ja toiminnallisesti itsenäisiä sekä toteutukseltaan ja toteuttajiltaan erilaisia, järjestelmien välisten integraatioiden manuaalinen testaus yhtenä kokonaisuutena on mahdollista suorittaa ainoastaan eduskunnan tuotantoympäristöä vastaavassa testiympäristössä, jossa kaikki eri tietojärjestelmät ovat asennettuina ja toiminnassa. Tietojärjestelmien välisten integraatioiden manuaalinen testaus vaatii oikeanlaisen testiympäristön lisäksi eduskunnan prosessien ja eduskunnan eri tietojärjestelmien toiminnan syvää osaamista. Tämän luvun alaluvuissa esitellään eduskunnan lainsäädäntötyön integroidut ydinjärjestelmät, näiden väliset integraatiot ja avainskenaariot sekä integraatioiden testaus.

4.1 Eduskunnan integroidut ydinjärjestelmät

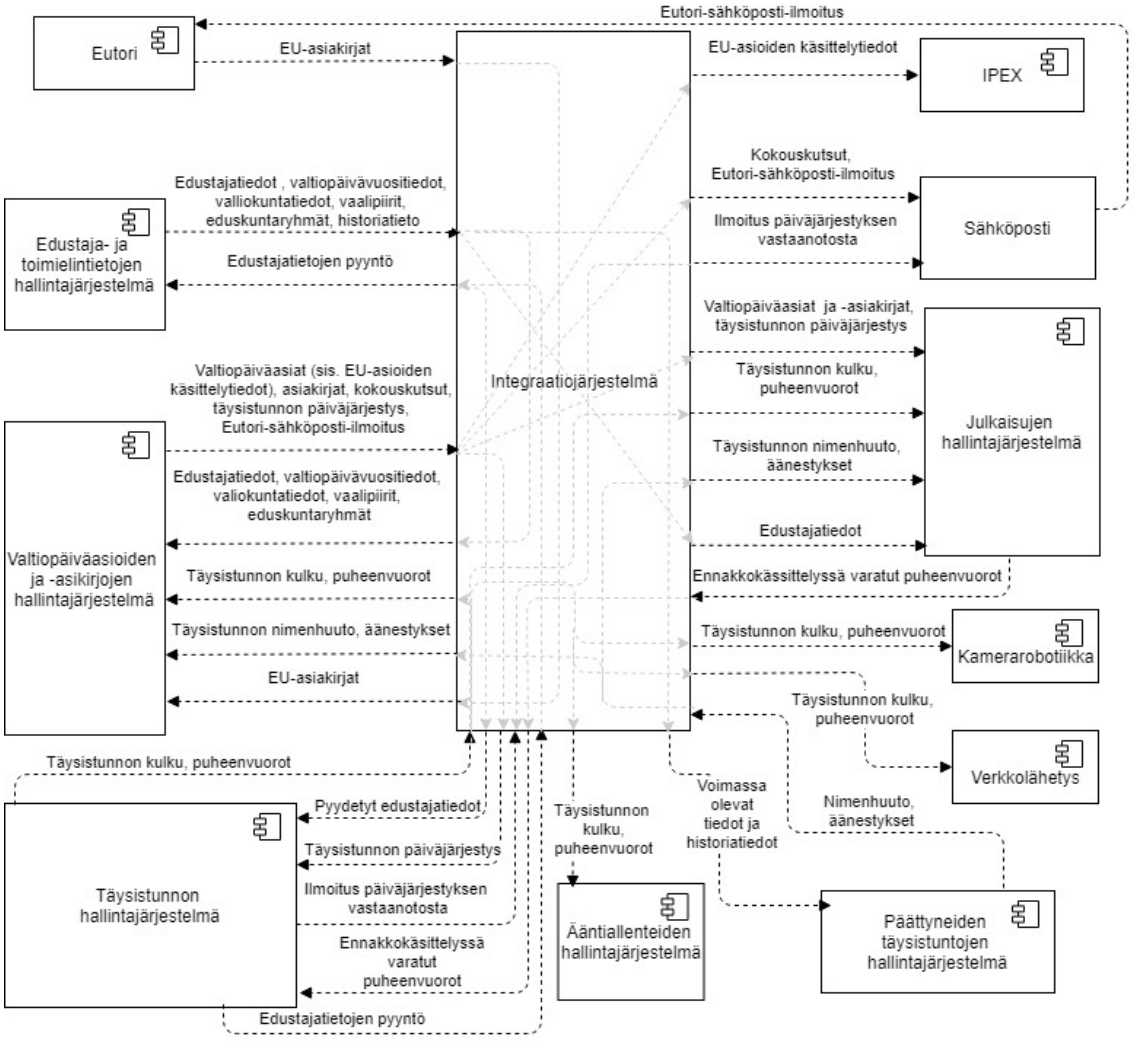
Jokainen eduskunnan ydinjärjestelmä on muista ydinjärjestelmistä riippumaton erillinen järjestelmä. Ydinjärjestelmien hallinta, kehitys sekä ylläpito on järjestetty järjestelmäkohtaisesti. Myös käyttäjäryhmät ovat järjestelmäkohtaisia.

Ydinjärjestelmät lähettävät tietoa integraatiojärjestelmälle, joka vastaanottaa tiedon ja välittää sen eteenpäin vastaanottavalle järjestelmälle siinä muodossa, missä vastaanottava järjestelmä pystyy sen vastaanottamaan. Ydinjärjestelmillä ei näin ollen ole ylläpidettävää suoraa liityntärajapintaa muiden ydinjärjestelmien tietotarpeita varten, vaan integraatiojärjestelmä toimii ydinjärjestelmien rajapintana. Integraatiojärjestelmä ei ota kantaa, mitä tietosisältöä ydinjärjestelmien lähettämät sanomat välittävät. Integraatiojärjestelmä ainoastaan validoi, tarvittaessa muokkaa ja lopulta reitittää tietoa sisältävät sanomat vastaanottaville järjestelmille.

Myös integraatiojärjestelmä on hallinnollisesti riippumaton muista ydinjärjestelmistä, mutta sen kehitys ja ylläpito puolestaan riippuu muiden ydinjärjestelmien integraatioista. Integraa-

tiojärjestelmän kehityksessä ja ylläpidossa on aina huomioitava jokainen integraatio ja integroituvan järjestelmän kehityssykli. Kun muutoksia tai ylläpitotöitä kuten ohjelmistopäivityksiä tehdään integraatiojärjestelmään, täytyy jokaisen liittymän toiminta varmistaa riittävän kattavin regressiotestein. Samoin jos tehdään muutoksia yhteen tai useampaan integroituun ydinjärjestelmään, on kaikkien näiden järjestelmien liittymät testattava mahdollisten regressiovirheiden varalta. Suurin osa eduskunnan integraatiojärjestelmän ylläpitotarpeista nousee muiden ydinjärjestelmien tarpeista kehittää tai lisätä uusia integraatioita muihin järjestelmiin.

Kuvassa 8 on kuvattu eduskunnan ydinjärjestelmien välinen vuorovaikutus ja integraatiot integraatiojärjestelmän kautta kulkevina tietovirtoina. Jokaista tietovirtaa varten on integraatiojärjestelmään perustettu oma integraationsa. Ydinjärjestelmien integraatioiden määrä integraatiojärjestelmässä on tällä hetkellä 29.



Kuva 8. Eduskunnan integroidut ydinjärjestelmät ja niiden integraatiot.

Ydinjärjestelmät vastaanottavat ja lähettävät sanomia myös eduskunnan ulkopuolisiin järjestelmiin integraatiojärjestelmäarkkitehtuurin tarjoaman ulkoisen rajapinnan kautta. Ydinjärjestelmiin integroidut eduskunnan ulkopuoliset järjestelmät ovat: Täysistunnon verkkolähetys, Täysistuntosalin kamerarobotiikka, EUTORI-järjestelmä (valtioneuvoston EU-asianhallintajärjestelmä) ja IPEX-järjestelmä (InterParliamentary EU information eXchange).

4.2 Järjestelmien väliset integraatiot ja avainskenaariot

Eduskunnan ydinjärjestelmien väliset integraatiot ja niiden tietovirtojen sisältö on kuvattu tarkemmin liitteessä 1. Alla on kuvattu lyhyesti kuvassa 8 esitetyt ydinjärjestelmät ja niiden väliset integraatiot.

Edustaja- ja toimielintietojen hallintajärjestelmä

Edustaja- ja toimielintietojen hallintajärjestelmä on integroitu valtiopäiväasioiden ja –asiakirjojen hallintajärjestelmään, täysistunnon hallintajärjestelmään, päättyneiden täysistuntojen hallintajärjestelmään sekä julkaisujen hallintajärjestelmään.

Edustaja- ja toimielintietojen hallintajärjestelmä on eduskunnan yksi vanhimmista käytössä olevista järjestelmistä. Järjestelmä lähettää eri sisältöistä tietoa muihin ydinjärjestelmiin. Edustaja- ja toimielintietojen hallintajärjestelmässä käynnistetään ajoja, jotka keräävät sen tietokannasta tietosisältöä xml-tiedostoihin. Jokaiselle integraatiolle ajetaan yksi tai useampi erisisältöinen xml-tiedosto riippuen vastaanottavan järjestelmän tietotarpeesta. Ajoja käynnistetään tietyin aikavälein sekä automaattisesti että käyttäjän toimesta järjestelmän käyttöliittymän kautta. Automaattiset ajot on ajastettu käynnistymään öisin.

Ajoihin liittyy paljon toiminnallisuutta jo ennen xml-sanomien lähetystä integraatiojärjestelmään, kuten ajon käynnistymisen ajastus, ajojen käynnistäminen, tietojen keruu ja xml-tiedostojen muodostus. Ajot ovat virhealttiita järjestelmäympäristössä tapahtuville häiriöille, jotka aiheuttavat ajoittain virheitä xml-sanomiin puutteellisena tietosisältönä. Tämä puolestaan vaikeuttaa virheen lähteen löytämistä, sillä virhe havaitaan vasta vastaanottavassa järjestelmässä käytön yhteydessä. Ajojen suuren määrän ja ajojen hallittavuuden vuoksi tehdyt ajastukset hidastavat liittymien testausta. Integraatiosta riippuen, käyttäjän käyttöliittymän kautta tilaaman siirtoajon lopputulosta joutuu odottamaan usein 40 minuutista yhteen vuorokauteen. Esimerkiksi edustajatietojen tilastoajot julkaisujen hallintajärjestelmään käynnistyvät vain öisin.

Valtiopäiväasioiden ja -asiakirjojen hallintajärjestelmä

Valtiopäiväasioiden ja –asiakirjojen hallintajärjestelmä on integroitu eduskunnan ydinjärjestelmien lisäksi kahteen eduskunnan ulkopuoliseen järjestelmään, Eutori-järjestelmään ja IPEX-järjestelmään. Valtiopäiväasioiden ja –asiakirjojen hallintajärjestelmä vastaanottaa edustaja- ja toi-

mielintietojenhallintajärjestelmältä valtiopäiviin liittyviä tietoja, kuten valtiopäivävuodet, edustajat ja valiokunnat. Täysistunnon hallintajärjestelmästä ja päättyneiden täysistuntojen hallintajärjestelmästä järjestelmä vastaanottaa täysistunnon kulkuun ja puheenvuoroihin liittyvät tiedot sekä nimenhuuto- ja äänestystiedot.

Valtiopäiväasioiden ja –asiakirjojen hallintajärjestelmä lähettää täysistunnon hallintajärjestelmään ja julkaisujen hallintajärjestelmään valtiopäiväasioiden käsittelytiedot ja valtiopäiväasiakirjojen sisältötiedot. Julkaisujen hallintajärjestelmä julkaisee valtiopäiväasiat ja –asiakirjat eduskunnan sisäisille ja ulkoisille verkkosivuille. Ulkoiset verkkosivut toimivat eduskunnan Avoin data –palvelun tietolähteenä.

Valtiopäiväasioiden ja asiakirjojen hallintajärjestelmä käsittää myös eri toimijoiden, kuten täysistunnon ja valiokuntien kokousten hallinnan. Järjestelmällä laaditaan ja ylläpidetään mm. täysistunnon päiväjärjestys ja pöytäkirja sekä valiokuntien kokoukset asiakirjoineen. Järjestelmä lähettää kokouksiin liittyvät kutsut integraatiojärjestelmälle, joka välittää kutsut edelleen sähköpostipalvelimelle lähetettäväksi.

Täysistunnon hallintajärjestelmä

Täysistunnon hallintajärjestelmä on integroitu kaikkiin eduskunnan ydinjärjestelmiin sekä eduskunnan ulkopuolisiin Kamerarobotiikka ja Verkkolähetys -järjestelmiin. Täysistunnon hallintajärjestelmä vastaanottaa edustajien ja toimielimien tietoja edustaja- ja toimielintietojen hallintajärjestelmästä, päiväjärjestyksen ja siinä olevien valtiopäiväasioiden tietoja valtiopäiväasioiden ja –asiakirjojen hallintajärjestelmästä sekä puheenvuorovaraustietoja julkaisujen hallintajärjestelmästä.

Täysistunnon hallintajärjestelmä lähettää muille järjestelmille tiedot täysistunnon kulusta, kuten alku- ja loppuaika, siirtymiset asiakohdasta toiseen, puheenvuoron alku- ja loppuaika ja puhujatiedot. Täysistunnon hallintajärjestelmä lähettää lisäksi kaikkiin vastaanottamansa päiväjärjestyksen sisältämiin asiakohtiin täysistunnossa päätetyt tiedot valtiopäiväasioiden ja –asiakirjojen hallintajärjestelmään, joka puolestaan lähettää valtiopäiväasioiden täysistunnon osalta päivitetyt käsittelytiedot julkaisujen hallintajärjestelmään, joka päivittää asiankäsittelytietojen verkkosivut.

Äänitallenteiden hallintajärjestelmä

Äänitallenteiden hallintajärjestelmällä on integraatio ainoastaan täysistunnon hallintajärjestelmään. Äänitallenteiden hallintajärjestelmä ei lähetä tietoja muihin järjestelmiin, vaan ainoastaan vastaanottaa täysistunnon hallintajärjestelmästä tulevat nk. lokaattoritiedot koskien täysistunnossa pidettyjä puheenvuoroja. Lokaattoritiedot auttavan nimensä mukaisesti äänitallenteiden hallintajärjestelmää äänitallenteiden sijoittamisen tehtävissä ja koskevat mm. puheenvuoron pitäjää ja puheenvuoron alku- ja loppumisaikatietoja.

Julkaisujen hallintajärjestelmä

Kaikki ydinjärjestelmät, jotka tuottavat eduskunnan verkkosivuilla julkaistavaa tietoa on integroitu julkaisujen hallintajärjestelmään. Julkaisusanomia järjestelmälle lähettävät edustaja- ja toimielintietojen hallintajärjestelmä, valtiopäiväasioiden ja –asiakirjojen hallintajärjestelmä, täysistunnon hallintajärjestelmä sekä päättyneiden täysistuntojen hallintajärjestelmä.

Julkaisujen hallintajärjestelmän integraatiot ovat yhtä lukuun ottamatta kaikki tiedon vastaanottamista varten. Julkaisujen hallintajärjestelmä lähettää tietoa ainoastaan täysistunnon hallintajärjestelmälle. Lähetettävä tieto käsittää täysistunnon ennakkokäsittelyaikana kansanedustajien sisäisten verkkosivujen palveluiden kautta varaamat puheenvuorot tulevaan täysistuntoon. Ennakkokäsittelyaika on kolme tuntia ennen täysistunnon alkua.

Edustaja- ja toimielintietojen hallintajärjestelmä lähettää julkaisujen hallintajärjestelmälle julkaisusanomat koskien nykyisten ja entisten kansanedustajien tietoja, kuten edustajan nimi, edustajatoimen alku- ja päättymisaika sekä edustajan jäsenyydet valiokunnissa. Valtiopäiväasioiden ja –asiakirjojen hallintajärjestelmän julkaisujen hallintajärjestelmälle lähettämät sanomat käsittävät valtiopäiväasioiden käsittelytietoja ja valtiopäiväasiakirjat, kuten täysistunnon päiväjärjestykset, valiokuntien esityslistat, valiokuntien lausunnot ja mietinnöt sekä täysistunnon pöytäkirjat. Julkaisujen hallintajärjestelmä käsittelee ja lukee näiltä järjestelmiltä vastaanottamansa sanomat ajastetusti sekä muuntaa nämä verkkosivuilla esitettävään muotoon.

Julkaisujen hallintajärjestelmä julkaisee täysistunnon hallintajärjestelmästä saamansa istunnon kulku ja puheenvuorotiedot täysistunto-verkkosivulla, joka näkyy eduskunnan sisäisillä ja ulkoisilla verkkosivuilla. Täysistunnon aikana tapahtuvista äänestyksistä saadaan tiedot päättyneiden täysistuntojen hallintajärjestelmästä ja näistä julkaisujen hallintajärjestelmä muodostaa omat sivunsa. Äänestyssivuille muodostetaan linkit täysistunto-sivulle niiden asiakohtien alle, joihin äänestykset kohdistuvat.

Päättyneiden täysistuntojen hallintajärjestelmä

Päättyneiden täysistuntojen hallintajärjestelmä on täysistunnon nimenhuuto- ja äänestystulosten hallintajärjestelmä sekä täysistunnon hallintajärjestelmän historiatietoa hallitseva järjestelmä. Järjestelmällä on integraatiot edustaja- ja toimielintietojen hallintajärjestelmään, valtiopäiväasioiden ja –asiakirjojenhallintajärjestelmään, täysistunnon hallintajärjestelmään sekä julkaisujen hallintajärjestelmään.

Järjestelmä suorittaa täysistunnon äänestysten ja nimenhuudon tuloslaskennan sekä lähettää näistä tuloksista sanomat valtiopäiväasioiden ja –asiakirjojen hallintajärjestelmään sekä julkaisujen hallintajärjestelmään. Päättyneiden täysistuntojen hallintajärjestelmä vastaanottaa edustaja- ja toimielintietojen hallintajärjestelmästä voimassa olevat tiedot sekä mm. historiatiedot valtiopäivävuosista, valiokunnista ja vaalipiireistä.

Avainskenaariot

Eduskunnan ydinjärjestelmien välisten integraatioiden taustalla on aina jonkin ydinjärjestelmän tarve saada tietoa. Järjestelmien välisten integraatioiden toiminnan testaamiseksi tulee ensin ymmärtää ydinjärjestelmien integraatioihin liittyvä toiminta. Integraatioiden kattavaan testaamiseen ei riitä tieto integraation kautta välitettävästä tietosisällöstä, vaan tietosisällön yhteys tietoa tuottavan ja tietoa käyttävän järjestelmän toimintaan on tärkeä ymmärtää. Järjestelmien järjestelmän testauksessa on tärkeää johtaa integraatiotestit niistä avainskenaarioista, joilla varmistetaan vuorovaikutus ympäristön kanssa ja osajärjestelmien välillä (Lima, 2018, s. 956). Skenaarioiden kuvaamisessa voi hyödyntää missiomallia, jossa järjestelmien järjestelmän mission ja osajärjestelmien tehtävien välinen yhteys sekä integraation laukaisevat tekijät käyvät hyvin selville.

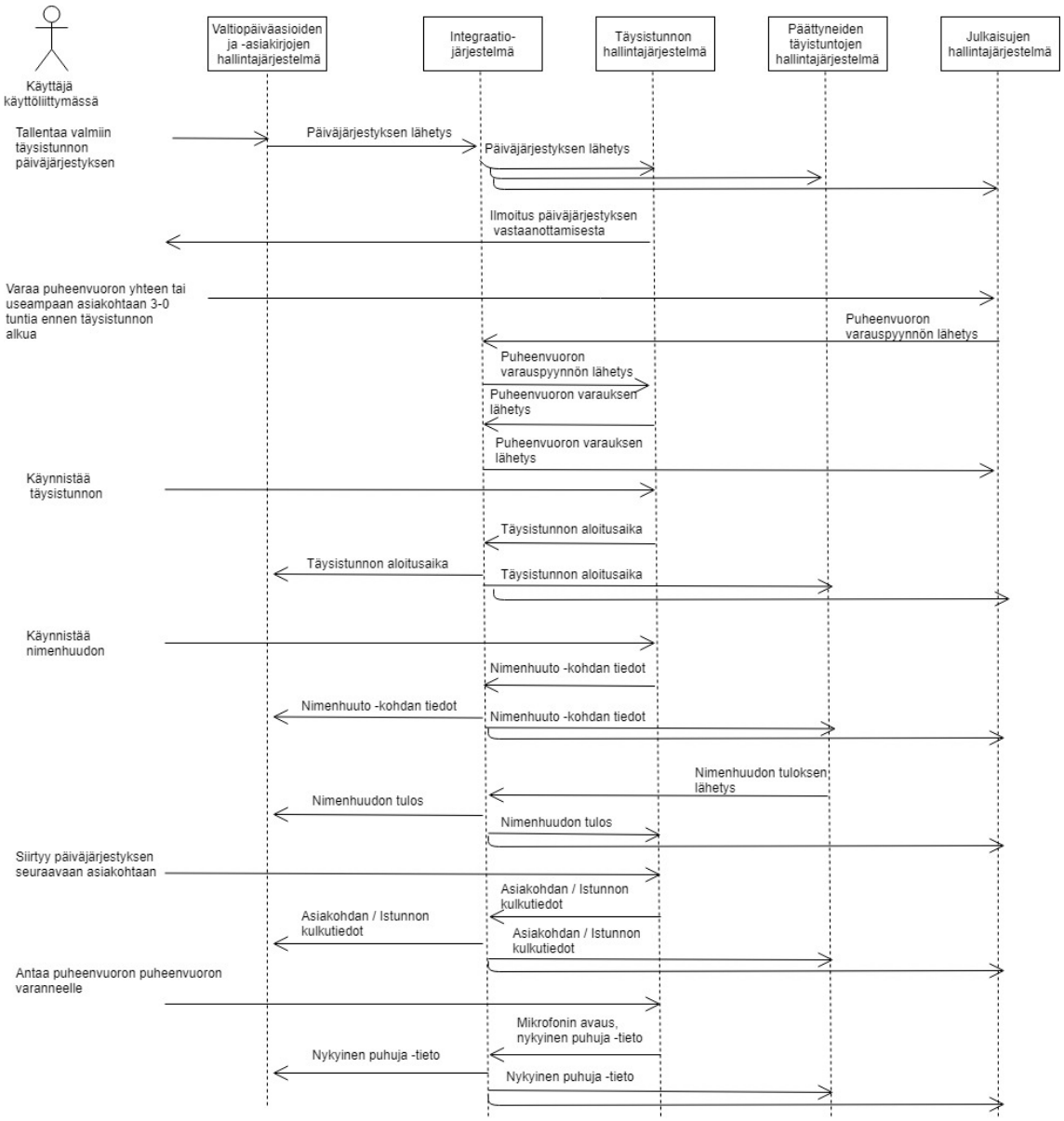
Eduskunnan tietojärjestelmien välisten integraatioiden avainskenaariot on johdettu ydinjärjestelmien niistä käyttötapauksista, joiden suorittaminen vaatii integraatioiden kautta noudettavia tai jaettavia tietoja. Avainskenaarioita ovat ne skenaariot, joissa eduskunnan eri ydinjärjestelmät toimivat yhdessä yhteisen mission saavuttamiseksi. Integraatiotestit ovat johdettu neljästä avainskenaariosta, jotka ovat kuvattu kuvissa 9 - 12. Näiden avainskenaarioiden sisältämä testijoukko sisältää kaikki ne testitapaukset, joissa edes kerran käydään jokainen yksittäinen järjestelmien välinen integraatio läpi.

Täysistunnon läsnäolotietojen ja puhujatietojen tuottaminen

Kuvassa 9 on esitetty avainskenaario, jossa valtiopäiväasioiden ja –asiakirjojen hallintajärjestelmä, täysistunnon hallintajärjestelmä, päättyneiden täysistuntojen hallintajärjestelmä ja julkaisujen hallintajärjestelmä toimivat yhdessä täysistunnon läsnäolotietojen ja puhujatietojen tuottamiseksi.

Täysistunnon läsnäolotietojen eli nimenhuudon tuloksen tuottaminen nimenhuudon päätyttyä päättyneiden täysistuntojen hallintajärjestelmästä on mahdollista, jos sitä ennen on käynnistetty nimenhuuto täysistunnon hallintajärjestelmässä. Nimenhuudon käynnistäminen puolestaan on mahdollista vasta, kun täysistunto on käynnistetty ja täysistunnon käynnistämisen laukaisevana tekijänä on täysistunnon päiväjärjestyksen saapuminen valtiopäiväasioiden- ja asiakirjojen hallintajärjestelmästä. Täysistunnon puhujatiedon tuottaminen täysistunnon hallintajärjestelmästä vaatii myös kuvassa näkyvien puheenvuoroa edeltävien käyttötapausten suorittamisen. Puheenvuoron varaaminen julkaisujen hallintajärjestelmän kautta ei ole välttämätöntä, mutta se on avainskenaariossa mukana integraation testaustarpeen vuoksi. Puheenvuoroja varataan myös täysistunnon hallintajärjestelmässä edustajan käyttöliittymän kautta. Integraatio julkaisujen hallintajärjestelmästä päin täysistunnon hallintajärjestelmään käsittää ainoastaan tämän yhden käyttötapauksen. Mikäli käyttötapaus ei olisi avainskenaariossa mukana, jäisi avainskenaarion pohjalta tehdystä integraatiotestauksesta testaamatta tämä integraatio.

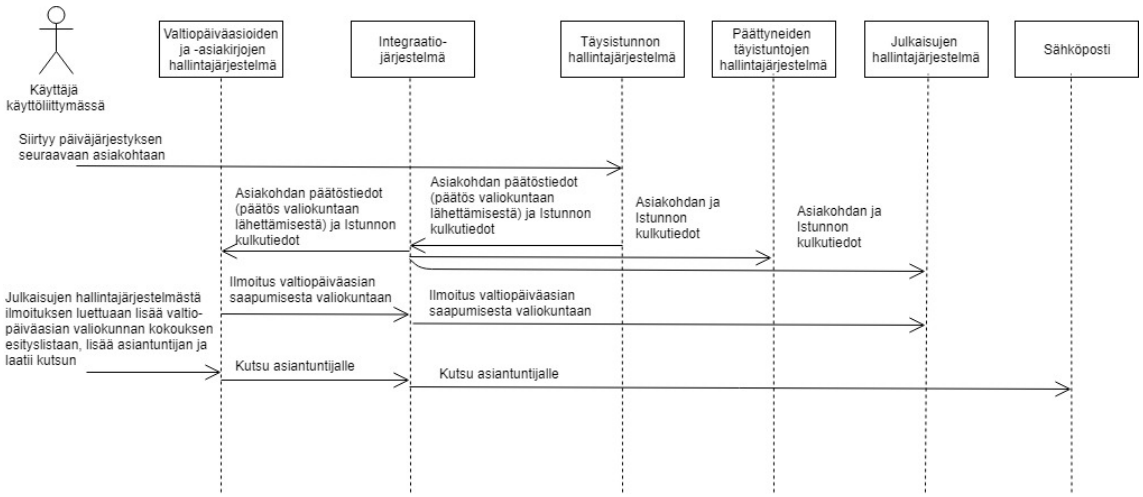
Avainskenaarion käyttötapaukset on suoritettava tietyssä järjestyksessä tiettynä aikana, jotta täysistunnon läsnäolotiedot ja puhujatiedot muodostuvat oikein. Integraatiotestauksessa on noudatettava samaa järjestystä erityisesti silloin, kun on tarpeen testata kaikki ydinjärjestelmien väliset integraatiot. Kuvasta 9 on rajattu pois palvelut eduskunnan ulkoisiin järjestelmiin kamerarobotiikkaan ja verkkolähetykseen. Ulkoiset järjestelmät saavat täysistunnosta ja sen kulusta samat tiedot kuin eduskunnan julkaisujen hallintajärjestelmä saa.



Kuva 9. Avainskenaario täysistunnon läsnäolotietojen ja puhujatietojen tuottamiseksi

Valiokunnan kokouskutsun lähettäminen

Kuvassa 10 on esitetty avainskenaario, jossa täysistunnon hallintajärjestelmä, valtiopäiväasioiden ja –asiakirjojen hallintajärjestelmä ja julkaisujen hallintajärjestelmä toimivat yhdessä mahdollistaakseen valiokunnan kokouskutsun lähettämisen asiantuntijalle.



Kuva 10. Avainskenaario valiokunnan kokouskutsun lähettämiseksi asiantuntijalle.

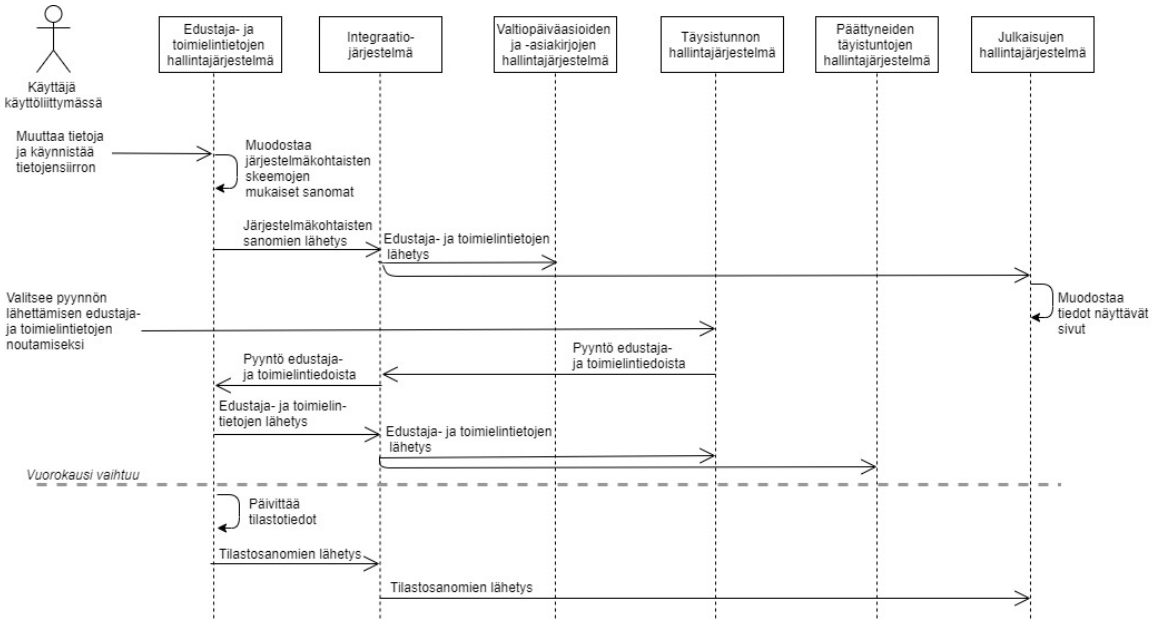
Avainskenaariota edeltää valtiopäiväasian lisääminen Valtiopäiväasioiden ja –asiakirjojen hallintajärjestelmään, valtiopäiväasian lisääminen täysistunnon päiväjärjestykseen, täysistunnon käynnistäminen ja täysistunnon hallintajärjestelmässä sellaisesta asiakohdasta siirtyminen seuraavaan asiakohtaan, jossa valtiopäiväasia on päätetty lähettää valiokuntaan. Valiokunta voi olla mikä tahansa eduskunnan 17 valiokunnasta. Kokouskutsun lähettäminen ei käytä eikä tarvitse integraatiota päättyneiden täysistuntojen hallintajärjestelmään, mutta integraatio on avainskenaariossa mukana havainnollistaakseen avainskenaarion yhdistymistä mm. kuvassa 9 sivulla 24 kuvattuun avainskenaarioon.

Edustaja- ja toimielintietojen jakaminen

Edustaja- ja toimielintietojen hallintajärjestelmän, valtiopäiväasioiden ja –asiakirjojen hallintajärjestelmän, täysistunnon hallintajärjestelmän, päättyneiden täysistuntojen hallintajärjestelmän ja julkaisujen hallintajärjestelmän yhteistoiminta edustaja- ja toimielintietojen hyödyntämisessä ja julkaisussa on esitetty kuvassa 11.

Edustaja- ja toimielintietoja hyödyntävien järjestelmien tulee saada tieto muuttuneista edustaja- ja toimielintiedoista. Tietojen siirto tapahtuu edustaja- ja toimielintietojen hallintajärjestelmästä osittain automaattisesti ja osittain manuaalisen tietojen siirron käynnistämisen jälkeen. Lähes kaikilla tietoja hyödyntävillä tai julkaisevilla järjestelmillä on tarpeidensa mukaiset skeemat tietojen toimittamiseksi. Skeemojen mukaiset xml-sanomat muodostetaan edustaja- ja toi-

mielintietojen hallintajärjestelmässä ja integraatiojärjestelmä toimittaa sanomat vastaanottaville järjestelmille. Täysistunnon ja päättyneiden täysistuntojen hallintajärjestelmille tiedot toimitetaan vain täysistunnon hallintajärjestelmien pyynnöstä. Muille järjestelmille edustaja- ja toimielintietojen hallintajärjestelmä toimittaa tiedot ajastetuin aikavälein. Tilastotiedoista, esim. naisedustajien ja miesedustajien määrä, muodostetaan ja lähetetään sanomat vain öisin. Avainskenaarion sisältämien integraatioiden testauksessa tulee huomioida edustaja- ja toimielintietojen hallintajärjestelmän sanomien muodostamisen ja lähetyksen ajastukset sekä tietoja vastaanottavien järjestelmien tietojen käsittelyyn liittyvät ajastukset.



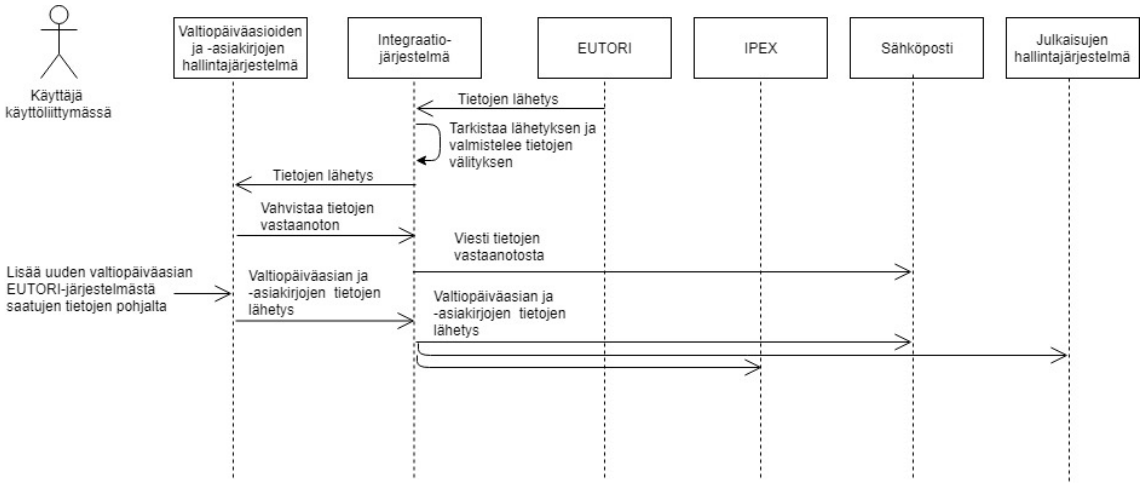
Kuva 11. Avainskenaario edustaja- ja toimielintietojen hyödyntämistä ja julkaisua varten.

Eduskunnan käsittelemien EU-asioiden käsittelytietojen jakaminen

Kuvassa 12 on esitetty avainskenaario, jossa eduskunnan valtiopäiväasioiden ja –asiakirjojen hallintajärjestelmä ja julkaisujen hallintajärjestelmä toimivat eduskunnan ulkopuolisten EUTORI- ja IPEX-järjestelmän kanssa eduskunnan käsittelyyn saapuneiden EU-asioiden käsittelytietojen jakamisessa.

Eduskunnan valtiopäiväasioiden ja –asiakirjojen hallintajärjestelmä vastaanottaa EUTORI-järjestelmästä tiedot eduskunnan käsittelyyn saapuvista EU-asioista. Käyttäjä lisää saatujen tietojen pohjalta oikeatyyppisen valtiopäiväasian valtiopäiväasioiden ja –asiakirjojen hallintajärjestelmään. Lisätyn asian tietojen pohjalta käynnistyy asian saaman eduskuntatunnuksen ja pysyvän verkkosivulinkin tietojen lähetys sähköpostitse EUTORI-järjestelmän määrittämälle vas-

taanottajalle. Asian käsittelytietojen julkaisua varten luotu pysyvä linkki eduskunnan verkkosivuille lähetetään myös IPEX-järjestelmään, josta kyseisen EU-asian eri EU-maiden parlamenttien käsittelytiedot ja asiakirjat julkaistaan. Pysyvä linkki muodostetaan valtiopäiväasioiden ja –asiakirjojen hallintajärjestelmässä ja julkaisujen hallintajärjestelmä muodostaa verkkosivun pysyvän linkin pohjalta.



Kuva 12. Avainskenaario eduskunnan käsittelemien EU-asioiden käsittelytietojen jakamisessa.

4.3 Järjestelmien välisten integraatioiden testaus

Järjestelmien välisten integraatioiden testaus suoritetaan järjestelmien testiympäristöissä. Jokaisen järjestelmän testiympäristö on tuotantoympäristöä vastaava, myös integraatiojärjestelmän. Järjestelmien testiympäristöt ovat integroitu keskenään täysin samaan tapaan kuin järjestelmien tuotantoympäristöt. Järjestelmiin ja erityisesti integraatiojärjestelmään kohdistuvien muutostöiden jälkeinen riittävän kattava integraatioiden testaus antaa täten riittävän varmuuden muutostöiden toimivuudesta ja asennuksesta tuotantoympäristöön.

Integraatiotestitapaukset on johdettu avainskenaarioista. Kaikkien eduskunnan ydinjärjestelmien välisten integraatioiden yhtäaikainen testaus vaatii kaikkien avainskenaarioiden sisältämien käyttötapausten suorittamisen oikeassa järjestyksessä oikeaan aikaan. Integraatioiden testaustarve ja testauksen laajuus riippuvat ydinjärjestelmään kohdistuvasta muutoksesta. Mikäli muutos kohdistuu vain yhteen ydinjärjestelmään, järjestelmien integraatiotestauksessa tulee testata regressiovirheiden varalta kaikki ne integraatiot, joiden kautta muut ydinjärjestelmät pyytävät tai tarjoavat kyseiselle järjestelmälle palveluja. Tällainen voi olla esimerkiksi julkaisujen hallintajärjestelmän versiopäivitys, jolloin on tarpeen testata kaikki ne integraatiot, joissa

julkaisujen hallintajärjestelmä on osallisena. Mikäli muutos kohdistuu kaikkiin ydinjärjestelmiin, kuten testi- ja tuotantoympäristön palvelinten päivitykset, on tarpeen testata kaikki järjestelmien väliset integraatiot.

Jokaiselle testauskierrokselle laaditaan testaussuunnitelma ja sovitaan testauksen järjestelyistä kaikkien testaukseen osallistuvien kanssa. Kriittisissä tuotantoympäristöä, integraatioita tai integraatiojärjestelmää koskevissa päivityksissä suunnitteluaikaa testaukselle ei ole. Tällöin testaus aloitetaan korkeimman prioriteetin testitapauksista edeten matalamman prioriteetin testitapauksiin testaukselle annetun ajan puitteissa. Testauspäällikkö varmistaa, että kaikki korkeimman prioriteetin testitapaukset tulee testattua riittävän kattavasti. Kaikki integraatiotestitapaukset on laadittu käytössä olevaan JIRA-järjestelmään.

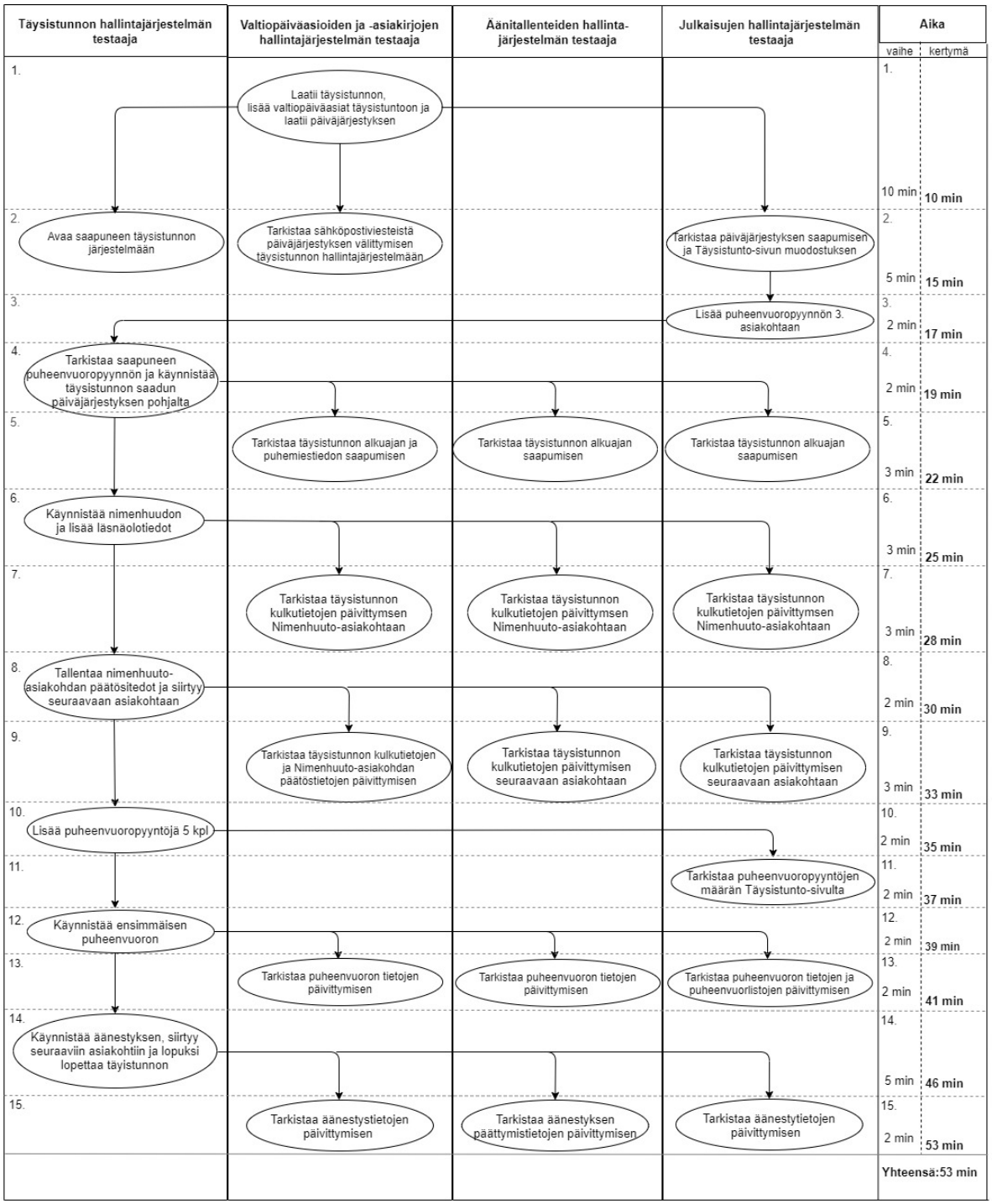
Integraatioiden testaus tapahtuu manuaalisesti avainskenaarioista johdettujen integraatiotestitapausten mukaisesti. Lähes jokaisen integraation testaus vaatii useamman ydinjärjestelmän käyttötapausten suorittamisen. Esimerkiksi täysistunnon hallintajärjestelmän integraatioiden testaus vaatii mm. päiväjärjestyksen tallentamisen valtiopäiväasioiden ja –asiakirjojen hallintajärjestelmässä ja puheenvuoron varaamisen julkaisujen hallintajärjestelmässä. Testaajat suorittavat avainskenaarioiden käyttötapaukset järjestelmien käyttöliittymien kautta ja tarkistavat järjestelmien käyttöliittymistä testauksen tuloksen. Testauksen tulos tarkistetaan myös integraatiojärjestelmän lokeilta, joihin tallentuvat tiedot integraatiojärjestelmän välittämistä sanomista sekä mahdollisista virhesanomista.

Täysistunnon hallintajärjestelmän integraatioiden testauksen kulku on esitetty kuvassa 13. Kuvassa integraatiotestauksen kulku on jaettu jokaisen testaukseen osallistuvan järjestelmän ja sen testaajan suorittamien testaustehtävien mukaisesti. Sarakkeilla on kuvattu testaukseen osallistuva järjestelmä ja riveillä on kuvattu testaustehtävien suoritusjärjestys ja testaustehtävän suoritukseen kuluva aika. Testaukseen yhteensä kuluva aika on kuvassa viimeisellä rivillä.

Kuvassa 13 riveillä 2, 5, 7, 9, 13 ja 15 kuvatut testaustehtävät voidaan suorittaa saman aikaisesti rinnakkain samalla rivillä kuvattujen testaustehtävien kanssa. Muilla riveillä kuvattuja testaustehtäviä ei voida suorittaa rinnakkain muiden testaustehtävien kanssa, vaan nämä testaustehtävät on suoritettava peräkkäin tietyssä järjestyksessä. Rivillä 15 kuvatut testaustehtävät testaavat päättyneiden täysistuntojen hallintajärjestelmän ja sarakkeilla esitettyjen järjestelmien väliset integraatiot, sillä äänestystiedot muodostuvat päättyneiden täysistuntojen hallintajärjestelmässä. Päättyneiden täysistuntojen hallintajärjestelmä osallistuu myös täysistunnon hallintajärjestelmän integraatioiden testaukseen, mutta se ei vaadi toimenpiteitä testaajilta vaan suorittaa nimenhuuto- eli läsnäolotietojen ja äänestystietojen laskennan automaattisesti. Tästä syystä järjestelmää ei ole lisätty omana sarakkeenaan kuvaan 13.

Täysistunnon hallintajärjestelmän integraatioiden laukaisevana tekijänä on päiväjärjestyksen

saapuminen valtiopäiväasioiden ja –asiakirjojen hallintajärjestelmästä sekä täysistunnon käynnistäminen tämän päiväjärjestyksen pohjalta. Kuvan 13 oikeanpuolimmaiseen sarakkeeseen on kuvattu testaukseen kuluva aika testaustehtävittäin sisältäen niin testaajien kuin järjestelmien tiedonvaihtoon kuluvan ajan. Testauksen suorittamiseen ensimmäisestä testaustehtävästä testaustulosten tarkastamiseen kuluu aikaa yhteensä hieman alle tunti, mikäli kaikki järjestelmät toimivat testauksessa ongelmitta.

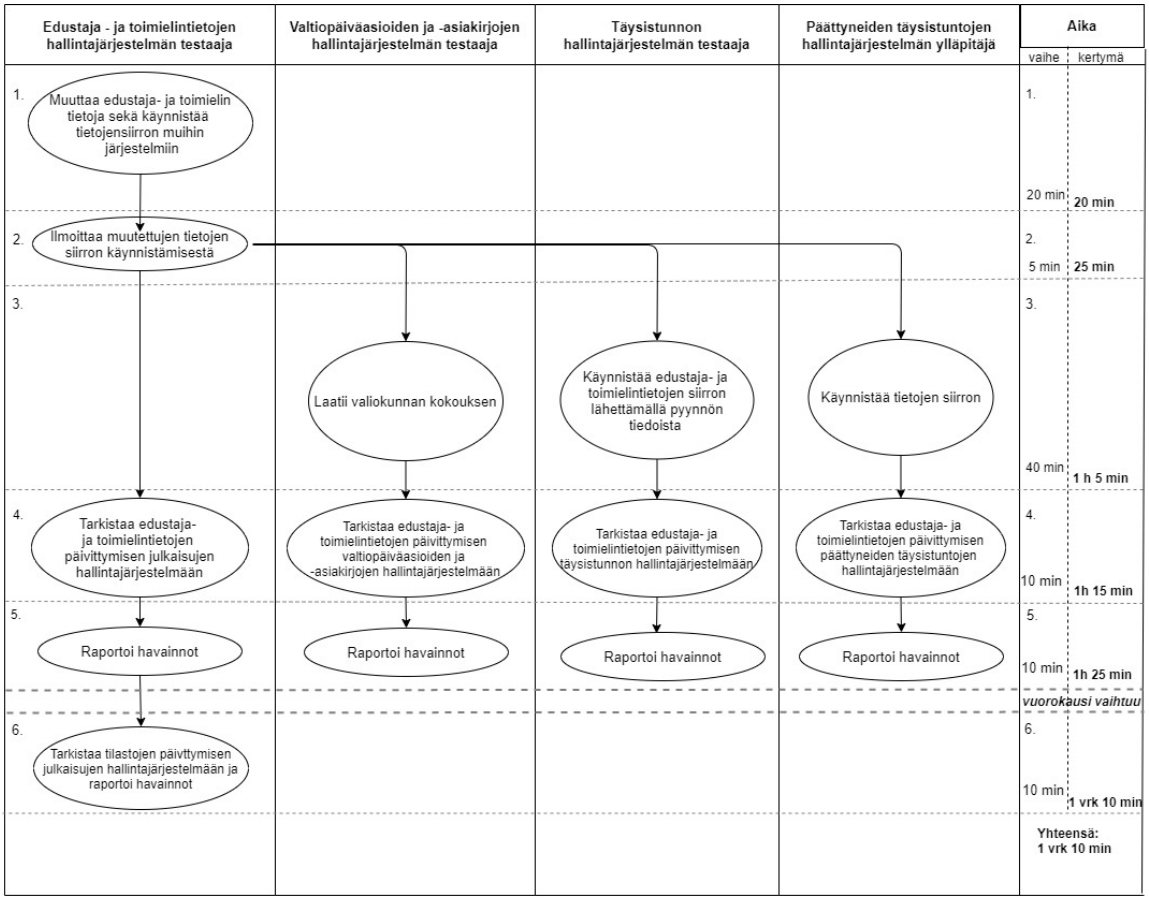


Kuva 13. Täysistunnon hallintajärjestelmän integraatioiden testauksen kulku

Järjestelmien integraatiot testataan tietyssä järjestyksessä, jotta kaikki integraatiot tulevat testattua vähintään kerran ja samaa integraatiota ei testattaisi useaan kertaan. Näin pyritään minimoimaan testien päällekkäisyyttä ja tehostamaan testauksen resurssienkäyttöä. Testitapauksissa lähettävän järjestelmän tietoihin pyritään tekemään sellainen päivitys, joka laukaisee sanomien lähetyksen kaikkiin tai mahdollisimman moniin kyseisen järjestelmän eri integraatioihin. Esimerkiksi edustaja- ja toimielintietojen hallintajärjestelmä lähettää edustajatietoja neljään eri järjestelmään. Tietojen lähetys kaikkiin neljään järjestelmään käynnistyy vain tiettyjen edustaja- ja toimielintietojen muutoksesta. Testauksessa muutetaan nämä tietyt tiedot järjestelmän käyttöliittymässä samalla kertaa, jotta edustajatietojen lähetys ja niitä välittävät kaikki integraatiot saadaan testattua mahdollisimman tehokkaasti yhdellä muutoskerralla. Kuvassa 14 on esitetty edustaja- ja toimielintietojen hallintajärjestelmän integraatioiden testauksen kulku ja testaukseen kuluva aika samaan tapaan sarakkeisiin ja riveihin jaettuna, kuten kuvassa 13 kuvattu täysistunnon hallintajärjestelmien integraatioiden testauksen kulku.

Kuvassa 14 kuvatut testaustehtävät riveillä 3 – 5 voidaan suorittaa saman aikaisesti rinnakkain. Rivillä 3 esitettyjen testaustehtävien testaukseen kuluva aika on laskettu hitaimman testattavan integraation mukaan, joka on edustaja- ja toimielintietojen päivittyminen julkaisujen hallintajärjestelmään. Testaustehtäviä riveillä 1, 2 ja 6 ei voida suorittaa rinnakkain muiden testaustehtävien kanssa. Testauksen suorittamiseen osallistuu testaajia kaikkiaan neljästä eri järjestelmästä. Integraatioiden laukaisevana tekijänä on tiettyjen tietojen muutos ja tiedonsiirron käynnistys edustaja- ja toimielintietojen hallintajärjestelmän käyttöliittymässä. Kuvan oikeanpuolimmaiseen sarakkeeseen on kuvattu testaukseen kuluva aika testaustehtävittäin, sisältäen tiedon muodostukseen ja vastaanottamiseen kuluva aika. Testauksen suorittamiseen ensimmäisestä testaustehtävästä testaustulosten tarkasteluun kuluu aikaa yhteensä yli vuorokausi.

Edustaja- ja toimielintietojen hallintajärjestelmän integraatioiden testauksen viimeinen tehtävä on tilastojen päivittymisen tarkistaminen ja se voidaan tehdä vasta seuraavana päivänä, koska järjestelmä muodostaa tilastot yöllä tapahtuvana eräajona. Tilastojen julkaisussa käytetään kuitenkin samaa integraatiota kuin kuvassa esitetyn tehtävän 4 edustaja- ja toimielintietojen julkaisussa, joten tilastojen testaus on integraatiotestauksen testijoukossa ainoastaan silloin, kun testataan erityisesti edustaja- ja toimielintietojen hallintajärjestelmän tai julkaisujen hallintajärjestelmän integraatioita. Mikäli integraatioita testataan palvelinpäivityksistä tai muihin ydinjärjestelmiin asennetuista kehitystöistä johtuen, ei tilastojen julkaisuja testata, vaan testaus lopetetaan 5. rivin testaustehtävien päättämiseen. Vähimmillään edustaja- ja toimielintietojen hallintajärjestelmän ja julkaisujen hallintajärjestelmän välisen integraation testaus vie 40 minuuttia. Aikaa kuluu muuttuneiden edustajatietojen tuottamiseen edustaja- ja toimielintietojen hallintajärjestelmässä ja näiden tietojen käsittelemiseen julkaisujen hallintajärjestelmässä.



Kuva 14. Edustaja- ja toimielintietojen hallintajärjestelmän integraatioiden testauksen kulku

Edustaja- ja toimielintietojen hallintajärjestelmä on tällä hetkellä vanhin käytössä olevista eduskunnan ydinjärjestelmistä. Sen integraatiot perustuvat erilaisiin asynkronisiin tiedonkeräyksiin ja tiedonsiirtoihin, joita käynnistetään sekä manuaalisesti käyttöliittymän kautta, että automaattisesti ajastettuina ajoina. Järjestelmien integraatiotestauksessa testaaja muuttaa ensin vaadittavat edustaja- ja toimielintiedot, jonka jälkeen hän voi käynnistää kaikki manuaalisesti käynnistettävät siirtotiedostojen ajot ja tietojen lähetykset muihin järjestelmiin. Siirtotiedostojen ajot eivät kuitenkaan lähetä tietoja täysistunnon hallintajärjestelmään ja päättyneiden täysistuntojen hallintajärjestelmään, vaan ne muodostavat siirrettävät tiedot sisältävät xml-tiedostot. Tuotantoympäristössä täysistunnon ollessa kesken ei täysistunnon hallintajärjestelmään ja päättyneiden täysistuntojen hallintajärjestelmään päivitetä edustaja- ja toimielintietoja mahdollisten häiriöiden välttämiseksi. Tästä syystä lähetys ei tapahdu automaattisesti ajon käynnistämisen jälkeen, vaan siirtotiedostot lähetetään vasta kun täysistunnon hallintajärjestelmästä tulee pyyntö tietojen lähettämiseen. Integraatiotestauksessa edustaja- ja toimielintietojen hallintajärjestelmän integraatioiden testaus suoritetaan ennen täysistunnon hallintajärjestelmän integraatioiden testausta, sillä edustaja- ja toimielintietoja ei voi päivittää myöskään testiympäristössä kesken täysistunnon.

Järjestelmien integraatiotestauksessa tulee huomioida kaikkien ydinjärjestelmien asettamat vaatimukset ja rajoitteet integraatioille. Integraatioiden kautta välitettävien sanomien rakenteen, tietosisällön ja ajoituksen tulee vastata ydinjärjestelmien asettamia vaatimuksia, jotta ydinjärjestelmät puolestaan toimivat oikein niille asetettujen vaatimusten mukaisesti.

4.4 Järjestelmien integraatiotestauksessa huomioitavat laatuvaatimukset

Järjestelmien järjestelmälle asetettujen vaatimusten kääntämisessä testattaviksi vaatimuksiksi tarvitaan laajaa järjestelmien rajojen ylittävää toimialan asiantuntijuutta (Ali, Petersen & Mäntylä, 2012, s.216). Eduskunnan järjestelmien järjestelmän integraatiotestauksessa huomioitavia laatuvaatimuksia ovat ydinjärjestelmille määritellyt integraatioita koskevat järjestelmäkohtaiset laatuvaatimukset sekä integraatiojärjestelmälle ohjelmistotuotteiden laatustandardin laatumallin ISO/IEC FDIS 25010:2010(E) laatuominaisuuksista johdetut laatuvaatimukset. Järjestelmäkohtaiset vaatimukset koskevat kyseisessä järjestelmässä vastaanotettavan ja lähetettävän sanoman lähetysprotokollaa, muotoa, rakennetta ja ajoitusta. Järjestelmäkohtaiset integraatioita koskevat vaatimukset on kuvattu kunkin yksittäisen järjestelmän liittymäkuvausdokumenteissa, joista ne ovat kirjattu testattaviksi vaatimuksiksi integraatiotestitapauksiin.

Ohjelmistotuotteiden laatustandardin laatumallin ISO/IEC FDIS 25010:2010(E) laatuominaisuuksista eduskunnan järjestelmien integraatiotestauksessa olennaisimmat ovat toiminnallinen oikeellisuus (Functional correctness), saatavuus (Availability), testattavuus (Testability) ja sovitettavuus (Adaptability) (ISO/IEC FDIS 25010:2010(E), 2010, s.10-16). Toiminnallisella oikeellisuudella viitataan tasoon, jolla järjestelmä tuottaa riittävän oikeanlaisia tuloksia (ISO/IEC FDIS 25010:2010(E), 2010, s.11). Eduskunnan järjestelmien järjestelmän integraatioiden toiminnallista oikeellisuutta arvioidaan tarkastelemalla yksittäisen integraation ja siihen osallistuvien osajärjestelmien toimintaa, esimerkiksi toimiiko täysistunnon päiväjärjestyksen lähetys oikein lähettävässä järjestelmässä, toimiiko päiväjärjestyksen vastaanottaminen ja välittäminen oikein integraatiojärjestelmässä ja toimiiko päiväjärjestyksen vastaanottaminen oikein vastaanottavassa järjestelmässä. Eduskunnan järjestelmien järjestelmän integraatioiden toiminnallinen oikeellisuus tulee olla tasolla, jossa kaikki toiminnot tuottavat oikeat tulokset järjestelmien järjestelmän mission tavoittamiseksi. Yksittäisen integraation kohdalla tämä tarkoittaa oikeanlaisen tiedon jakamista oikealla tavalla oikeaan aikaan integraatioon osallistuvien osajärjestelmien välillä.

Saatavuudella viitataan tasoon, jolla järjestelmien järjestelmä ja sen osajärjestelmät ovat toiminnassa ja käytettävissä. Eduskunnan järjestelmien järjestelmän kattava integraatiotestaus edel-

lyttää kaikkien järjestelmien saatavuutta. Järjestelmien järjestelmän ja sen kaikkien osajärjestelmien on oltava saatavilla arkisin klo 07.00 – 24.00 välisen ajan. Lisäksi järjestelmien on oltava saatavilla aina täysistuntojen ollessa käynnissä. Täysistunnot saattavat kestää yli yön, joten kaikkien järjestelmien tulee tarvittaessa olla saatavilla usean vuorokauden ajan ilman katkoja. Järjestelmien järjestelmän tulee olla reaaliajassa monitoroitavissa ja hallittavissa. Integraatiotestauksessa järjestelmien järjestelmää monitoroidaan ja hallitaan järjestelmien ylläpitäjien toimesta. Mahdollisten häiriöiden aiheuttamien katkojen varalta järjestelmien järjestelmän toiminta voidaan ohjata toiseen identtiseen toimintaympäristöön. Järjestelmien järjestelmän testiympäristö on tuotantoympäristön tapaan kahdennettu ja toiminnan ohjaaminen toiseen toimintaympäristöön testataan testiympäristössä niin kutsuttujen korkean saatavuuden (High Availability) testitapauksin. Integraatioiden toiminta testataan korkean saatavuuden testitapauksissa siten, että toimintaympäristö vaihdetaan kesken testitäysistunnon kulun, jolloin sanomia kulkee jatkuvasti suuri määrä lähes kaikkien osajärjestelmien välillä. Järjestelmien järjestelmän tulee varmistaa integraatioiden häiriötön toiminta ja hoitaa sanomien käsittely siten, että yksikään sanoma ei jää käsittelemättä tai muutu käsittelyn aikana aiheuttaen jonkin osajärjestelmän toimintaan häiriöitä. Sanomien eheys varmistetaan integraatiotestauksessa sanoman vastaanottavan osajärjestelmän toiminnan testaamisella.

Testattavuudella viitataan tasoon, kuinka tehokkaasti järjestelmälle kyetään määrittelemään testauskriteerit ja suorittamaan testit näiden kriteerien täyttymisen arvioimiseksi (ISO/IEC FDIS 25010:2010(E), 2010, s.15). Testattavuuden tulee olla korkealla tasolla, sillä järjestelmien järjestelmän integraatiotestauksella varmistetaan kaikkien järjestelmien yhteistoiminta yhteisen mission tavoittamiseksi. Kaikki integraatiot tulee olla mahdollista testata osajärjestelmien käyttäjien toimesta ja testaajina toimivat käyttäjät voivat arvioida itse testauksensa tulokset. Testiympäristön tulee toimia tuotantoympäristön kaltaisesti. Testattavuuden korkean tason saavuttamiseksi integraatiotestauksen kriteerit ja testiaskeleet on johdettu avainskenaarioista ja järjestelmäkohtaisista vaatimuksista. Integraatiotestauksessa käyttäjät suorittavat toiminnot osajärjestelmissä samaan tapaan, kuin ne suoritettaisiin tuotantotilanteessa tuotantoympäristössä. Integraatiojärjestelmän tulee kirjoittaa tapahtumalogia, virhelogia ja suorituskykyraporttia, joita käytetään apuna testauskriteerien täyttymisen arvioinnissa. Logien ja suorituskykyraportin sisältö käydään läpi integraatiotestauksen lopuksi.

Sovitettavuus viittaa tasoon, kuinka tehokkaasti järjestelmä on sovitettavissa erilaisiin tai kehitteillä oleviin laitteisto-, ohjelmisto- tai muihin toiminta- ja käyttöympäristöihin (ISO/IEC FDIS 25010:2010(E), 2010, s.15). Korkea saatavuus edellyttää eduskunnan järjestelmien järjestelmän nopeaa sovittamista häiriöiden varalla olevaan toimintaympäristöön. Korkean saatavuuden testeissä toimintaympäristön palvelut katkaistaan kesken palveluiden käytön, kuten häiriön sattuessa tapahtuu, ja siirrytään käyttämään häiriöiden varalla olevaa ympäristöä. Integraatiotestaus ei sisällä korkean saatavuuden testausta, mutta korkean saatavuuden testaus sisältää aina

integraatiotestauksen, jossa pääpaino on sanomien eheyden varmistamisessa. Sisäisen kapasiteetin skaalautuvuus (esim. näyttöjen kentät, taulukot, tapahtumamäärät, ym.) sisältyy sovitettavuuteen (ISO/IEC FDIS 25010:2010(E), 2010, s.15). Eduskunnan integraatiojärjestelmän tuottamassa suorituskykyraportissa tulee näkyä sanomien lukumäärä sekä sanomien käsittelyjonojen läpimenoajat. Integraatiotestauksen lopuksi integraatiojärjestelmän sisäisen kapasiteetin skaalautuvuutta arvioidaan suorituskykyraportin sisällön avulla.

Eduskunnan järjestelmien järjestelmän integraatiotestauksessa huomioitavat laatuvaatimukset on kirjattu integraatiojärjestelmälle asetettujen vaatimusten taulukkoon. Alla olevassa taulukossa 2 on listattu edellä mainitut järjestelmien järjestelmälle oleellisista laatuominaisuuksista johdetut laatuvaatimukset.

Integraatioalustaratkaisulle asetettu vaatimus	Liittymien testauksessa huomioitava myös vastaanottotarkastuksen ja käyttöönoton jälkeen	Miten huomiodaan testauksessa
Vaatimus 1.6: Järjestelmän tulee kirjoittaa tapahtumalogia	kyllä	Testauksen päätyttyä tarkistetaan
Vaatimus 1.7: Järjestelmän tulee kirjoittaa virhelogia	kyllä	Testauksen päätyttyä tarkistetaan
Vaatimus 1.10: Järjestelmän tulee tuottaa suorituskykyraportit (jonojen läpimenoajat, sanomamäärät, yms.)	kyllä	Testauksen päätyttyä tarkistetaan
Vaatimus 1.11: Järjestelmän tulee olla reaaliajassa monitoroitavissa ja hallittavissa	kyllä	Testauksen aikana monitoroidaan
Vaatimus 1.17: Järjestelmän tulee välittää viestit muuttumattomina (eheys)	kyllä	Tarkistetaan vastaanottavassa järjestelmässä

Taulukko 2. Integraatiotestauksessa huomioitavat integraatiojärjestelmälle asetetut vaatimukset.

Taulukossa keskimmäinen sarake tarkoittaa sitä, että vaatimus on huomioitava jokaisessa integraatiotestauksessa järjestelmän elinkaaren ajan. Integraatiojärjestelmälle on määritetty integraatiotestauksessa huomioitavien vaatimusten lisäksi myös muita ja muissa vaiheissa huomioitavia vaatimuksia, joita ei ole listattu taulukkoon 2. Taulukon viimeisessä sarakkeessa kuvataan lyhyesti, kuinka vaatimus huomioidaan integraatiotestauksessa. Testauksen hallinnan työkalun avulla vaatimukset saadaan tuotua testitapauksien suunnitteluun ja jokaisen vaatimuksen testauksen tilannetta voidaan seurata ja jäljittää (Spillner, Linz, & Schaefer, 2014, s. 286). Eduskunnan järjestelmien integraatiotestauksessa huomioitavat vaatimukset on kirjattu testauksen hallinnan työkalulla, JIRA-järjestelmällä, laadittuihin testitapauksiin.

5 Eduskunnan järjestelmien integraatiotestauksen haasteet

Eduskunnan järjestelmien integraatiotestauksen suurimmat haasteet ovat testauksen suunnitteluun, koordinointiin ja itse testaukseen kuluva aika sekä testauksen vaatimat henkilöresurssit. Jokainen järjestelmä on toiminnallisesti ja hallinnollisesti itsenäinen, jolloin jokaisella järjestelmällä on mm. oma kehitystiiminsä, omat testaajansa sekä omat kehityssyklinsä. Vaikka järjestelmät ovat toiminnallisesti ja hallinnollisesti itsenäisiä, tarvitsevat järjestelmät palveluita toisiltaan saavuttaakseen yhteisen tavoitteen lainsäädäntötyön prosessien tukemisessa. Esimerkiksi julkaisujen hallintajärjestelmä voi muodostaa täysistunnon puhujatiedot näkyviin eduskunnan verkkosivuille vasta, kun se on vastaanottanut puhujatiedot sisältävän sanoman, jonka täysistunnon hallintajärjestelmä on lähettänyt.

Palveluiden voimakkaiden riippuvuuksien vuoksi järjestelmien kehityssyklit on pyrittävä synkronoimaan. Jokaisen kehityssyklin testauksen suunnittelussa tulee kiinnittää huomiota testitapausten priorisointiin sekä siihen, milloin tarvittavat muiden järjestelmien tarjoamat palvelut ovat saatavissa, jotta testaus ylipäätään on mahdollista suorittaa. Esimerkiksi julkaisujen hallintajärjestelmään tehtyjen muutosten jälkeen tulee korkealla prioriteetilla testata kaikki julkaisujen hallintajärjestelmän integraatiot ja hyvissä ajoin ennen testausta varmistaa näiden integraatioiden kautta palveluita tarjoavien ja käyttävien järjestelmien saatavuus testauksen suunniteltuna ajankohtana. Kaikki integraatiot kattava järjestelmien integraatiotestaus on suunniteltava viikkoja etukäteen, jotta kaikkien eri järjestelmien saatavuudesta testiympäristössä sekä testaajien osallistumisesta tiettynä ajankohtana voidaan sopia. Kiireellisissä päivitysasennuksissa suunnitteluaikaa on hyvin vähän tai ei ollenkaan, minkä seurauksena kaikkien integraatioiden kattava testaus manuaalisesti suorittamalla on mahdotonta.

5.1 Testauksen resurssit

Integraatiotestauksessa testitapaukset suoritetaan järjestelmien käyttöliittymien kautta, ja myös testauksen tulos tarkistetaan järjestelmien käyttöliittymistä. Tämä vaatii testaajalta järjestelmän ja sen käyttöliittymän syvää osaamista. Järjestelmiin ja järjestelmien käyttöliittymiin kohdistuu myös jatkuvaa kehitystyötä, jonka seurauksena toiminnot käyttöliittymissä muuttuvat aika ajoin. Testaajat tuntevat usein yhden tai kahden järjestelmän toiminnot riittävän hyvin, minkä vuoksi järjestelmien integraatioiden luotettavaan testaukseen tarvitaan testaajia jokaisesta eri järjestelmästä. Integraatiotestaus vaatii kaikkiaan kahdeksan eduskunnan testaajan, yhden testauskoordinaattorin ja yhden integraatiojärjestelmän ylläpitäjän työtä. Eduskunnan testaajat ovat pääsääntöisesti järjestelmien pääkäyttäjiä, jotka suorittavat testausta oman lainsäädäntöprosessiin liittyvän työnsä ohella. Integraatiojärjestelmän ylläpitäjä seuraa testauksen

aikana integraatiojärjestelmässä kulkevien sanomien vastaanottoa ja lähetystä sekä seuraa integraatiojärjestelmän tapahtuma- ja virhelokeihin kertyvää tietoa.

Integraatiotestauksessa hitainta on testata edustaja- ja toimielintietojen hallintajärjestelmän ja valtiopäiväasioiden ja –asiakirjojen hallintajärjestelmän kaikki integraatiot. Tämä johtuu siitä, että molempien järjestelmien tilastot muodostuvat yöllä tapahtuvissa eräajoissa ja tilastojen oikeanlainen päivittyminen voidaan tarkistaa eduskunnan verkkosivuilta vasta seuraavana päivänä. Integraatiotestaukseen kuluva aika riippuu testijoukosta. Mikäli kaikkien järjestelmien kaikki integraatiot ovat tarpeen testata, testaukseen kuluu aikaa aina vähintään yksi vuorokausi. Mikäli tilastoja koskevat integraatiotestit voidaan jättää suorittamatta, kuten palvelinpäivitysten jälkeisessä integraatiotestauksessa voidaan, integraatiotestaukseen kuluu vähintään 2 tuntia 25 minuuttia. Luvussa 4.3 kuvatut tietyssä järjestyksessä tiettyyn aikaan suoritettavat integraatiotestit asettavat manuaaliselle integraatiotestaukselle tämän vähimmäisajan. Ennen täysistunnon hallintajärjestelmän integraatioiden testausta pitää edustaja- ja toimielintietojen hallintajärjestelmän integraatioiden testaustehtävistä olla suoritettu kaksi ensimmäistä: edustaja- ja toimielintietoihin sovitun muutoksen tekeminen ja tiedonsiirtojen käynnistys sekä ilmoitus muille testaajille siirtojen käynnistymisestä. Ne vievät aikaa 25 minuuttia, ja niiden tulee olla tehtynä, jotta täysistunnon hallintajärjestelmän testaaja voi käynnistää edustaja- ja toimielintietojen siirron täysistunnon hallintajärjestelmään ennen sen integraatioiden testausta ja täysistunnon aloittamista. Täysistunnon hallintajärjestelmän integraatioiden testaus sen edellyttämä testausaineiston luonti mukaan laskettuna vie aikaa vähintään 2 tuntia. Samaan aikaan voidaan rinnakkain testata loput muut integraatiot.

Integraatiotestauksen manuaaliseen suorittamiseen kuluu aikaa paitsi itse testaamiseen myös tiedonvälitykseen eri testaajien välillä. Palveluiden välillä olevien riippuvuuksien vuoksi monet integraatiotestauksen testaustehtävät joudutaan suorittamaan tietyssä järjestyksessä. Esimerkiksi edustaja- ja toimielintietojen hallintajärjestelmän testaajan täytyy ilmoittaa testauskoordinaattorille ja muiden järjestelmien testaajille tietoihin tehdyistä muutoksista ja testaustyönsä valmistumisesta, jotta muut testaajat voivat suorittaa omat testaustehtävänsä muissa järjestelmissä kyseisten muutettujen tietojen pohjalta ja sen jälkeen ilmoittaa testaustulokset puolestaan edustaja- ja toimielintietojen hallintajärjestelmän testaajalle sekä testauskoordinaattorille. Myös esimerkiksi täysistunnon hallintajärjestelmän ja julkaisujen hallintajärjestelmän välinen integraatio voidaan testata vain ja vasta, kun valtiopäiväasioiden ja –asiakirjojen hallintajärjestelmän ja täysistunnon hallintajärjestelmän välinen integraatio on onnistuneesti testattu täysistunnon päiväjärjestyksen lähetyksellä.

Erityisesti täysistunnon hallintajärjestelmän integraatioita testatessa tulee kaikkien testaajien, testauskoordinaattorin ja integraatiojärjestelmän ylläpitäjän tietää tarkasti testaustehtävien

eteneminen ja testaajien havainnot, esimerkiksi tuliko päiväjärjestys täysistunnon hallintajärjestelmään, onko kokous siirretty oikeaan vaiheeseen täysistunnon aloittamiseksi, tuliko täysistunnossa pidetty puheenvuoro äänitallenteiden hallintajärjestelmään ja tuliko äänestyksen tiedot julkaisujen hallintajärjestelmään. Mikäli jokin integraatio ei toimi tai toimii virheellisesti, pyritään virhe korjaamaan tai vähintään selvittämään virheen mahdollinen syy ennen testauksen jatkamista. Täysistunnon hallintajärjestelmän integraatioiden testaus suoritetaan yhdessä testaajien ollessa samassa tilassa, fyysisessä tai virtuaalisessa, jotta tiedonvälitykseen kuluva aika saadaan minimoitua ja kaikki testaajat voivat reaaliajassa keskustellen jakaa tietoa testauksen kulusta.

Testauksen vaatimat henkilöresurssit, aika ja yhteisen testauksen tarve ovat haasteita myös testauksen suunnittelulle. Integraatiotestauksen suunnittelussa tulee huomioida myös järjestelmien ylläpitäjien aikataulu riippuen integraatiotestauksen tarkoituksesta eli testataanko palvelinpäivityksiä, sovelluspäivityksiä vai testataanko jonkin tietyn ydinjärjestelmän kehitystöiden asennuksia. Testauksen ajankohdan tulee sopia sekä testaajille että ylläpitäjille ja lisäksi huomioida kaikkien testattavien järjestelmien omat kehityssyklit. Sopivan ajankohdan löytämiseen ja testauksen suunnitteluun kuluu testauspäälliköltä vähintään 2 tuntia, edellyttäen testijoukon pysyvän pitkälle samanlaisena, kuin aiemmin vastaavassa integraatiotestauksessa on ollut. Integraatiotestauksen suunnitteluun ja suoritukseen kuluu aikaa yhteensä vähintään 4 tuntia. Koska testaajat testaavat järjestelmiä varsinaisen työnsä ohessa ja tarvittavien testaajien määrä on suuri, on testauksen ajankohta ja testaukseen osallistuvat testaajat varattava ja sovittava useampi viikko ennen testausta. Kiireellisten tietoturvapäivitysten osalta järjestelmien integraatiotestauksen suunnitteluaika ei ole, ja siksi järjestelmien integraatioita ei aina saada asennuksen jälkeen testattua riittävän kattavasti, mikä lisää riskiä virheiden läpipääsystä tuotantoympäristöön.

5.2 Testauksen suoritus

Palveluiden voimakkaiden riippuvuuksien vuoksi testaus on lähes kaikkien ydinjärjestelmien välisten integraatioiden osalta suoritettava tietyssä järjestyksessä, kuten on kuvattu luvussa 4.3. Tämä edellyttää integraatiotestaukselta tarkkaa suunnittelua ja koordinointia sekä testaustehtävien suorittamista tarkasti sovittujen testiaskeleiden mukaisesti. Mikäli testaustehtävien suorituksessa tapahtuu virhe palveluita tarjoavassa tai käyttävässä järjestelmässä, saattaa tämä johtaa väärään positiiviseen (false positive) virhehavaintoon järjestelmien integraatiotestauksessa.

Palveluiden voimakkaista riippuvuuksista huolimatta jotkin integraatiotestauksen testitapaukset voidaan suorittaa rinnakkain. Nämä testitapaukset testaavat valtiopäiväasioiden ja –asiakirjojen hallintajärjestelmän integraatiota sähköpostiin ja eduskunnan ulkopuolisiin järjestelmiin.

Valiokunnan kokouskutsun lähettäminen ei ole riippuvainen muiden ydinjärjestelmien tarjoamista palveluista, ja se voidaan testata integraatiotestauksessa missä vaiheessa tahansa. Myöskään EUTORI-järjestelmästä saatu palvelu ja IPEX-järjestelmälle tarjottava palvelu eivät riipu muiden ydinjärjestelmien tarjoamista palveluista, ja nämä voidaan testata rinnakkain muiden testitapausten kanssa. Testauksen edellytyksenä on testaussuunnitteluvaiheessa EUTORI-järjestelmän testaajalle etukäteen lähetetyn aineistopyynnön mukaisen aineiston saapuminen EUTORI-järjestelmästä integraatiojärjestelmään. Integraatiojärjestelmä välittää EUTORI-järjestelmästä saapuneen aineiston valtiopäiväasioiden ja –asiakirjojen hallintajärjestelmään. Mikäli sähköposti-, EUTORI- ja IPEX-integraatiot testataan rinnakkain, testaus vaatii kahden testaajan työn 10 minuutin ajan. Mikäli nämä testataan peräkkäin, testaus vaatii yhden testaajan työn 20 minuutin ajan. Näiden integraatioiden testaus suoritetaan usein samanaikaisesti täysistunnon hallintajärjestelmän integraatioiden testauksen kanssa.

5.3 Toimintaympäristön vaikutus

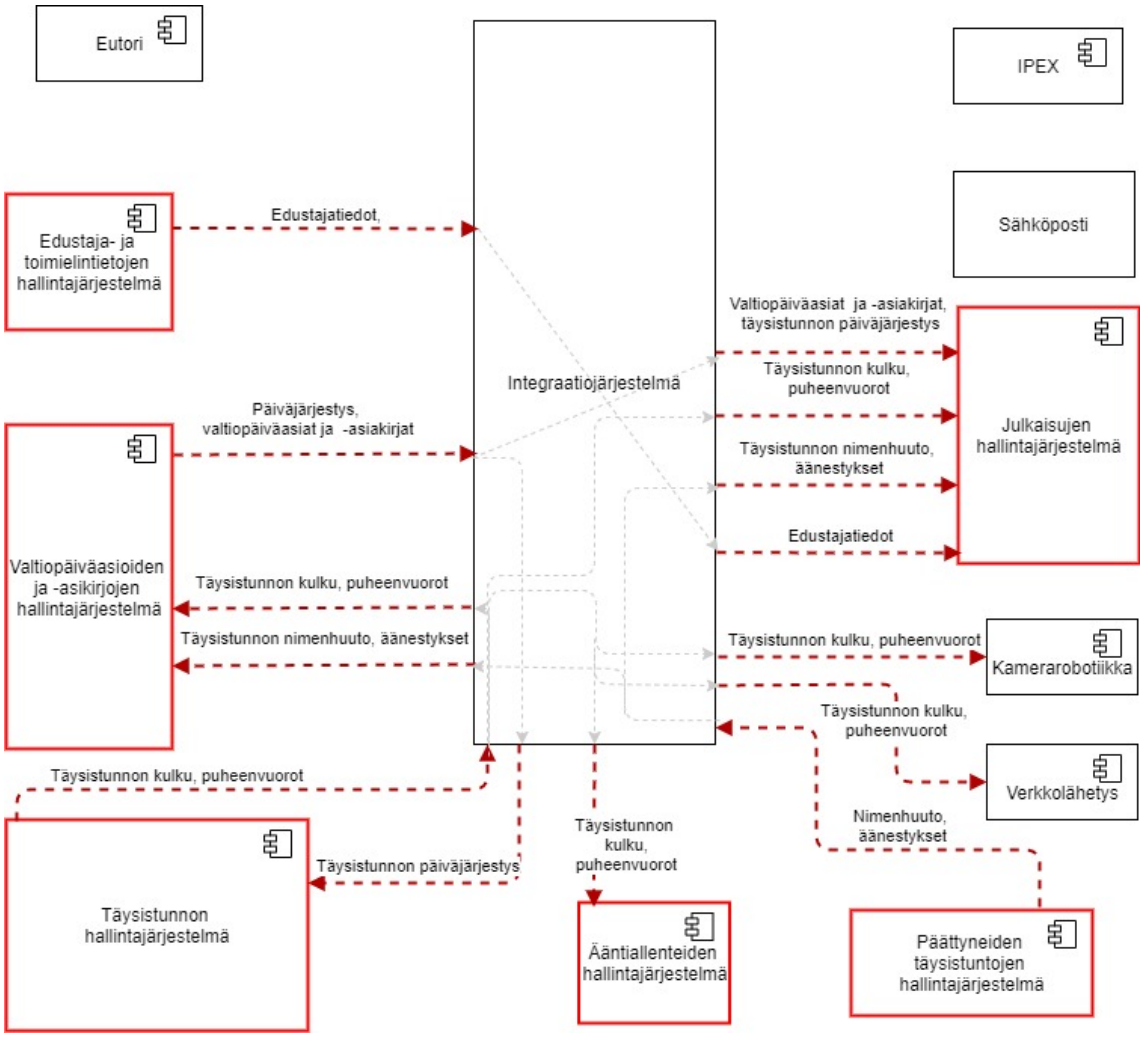
Kaikilla lainsäädäntötyön tietojärjestelmillä on eduskunnassa tuotantoympäristöä vastaava testausympäristö. Järjestelmien toimittajilla ja ylläpitäjillä on omat kehitysympäristönsä, joissa he voivat testata kehittämänsä ja ylläpitämänsä järjestelmän integraatioita mm. hyödyntämällä SoapUI -työkalulla laadittuja todellisten integraatioiden välittämien palveluiden jäljitelmiä (SoapUI, 2020). Eduskunnan testiympäristö on ainoa ympäristö, jossa järjestelmien välisiä integraatioita on mahdollista testata päästä päähän avainskenaarioista johdettujen testitapausten mukaisesti. Eduskunnan testaajat suorittavat järjestelmien integraatiotestauksen, sillä integraatiotestaus vaatii toimintaympäristön, järjestelmien toiminnan ja avainskenaarioiden hyvää osaamista.

Järjestelmien itsenäisyydestä ja monitoimittajaympäristöstä aiheutuvia haasteita integraatiotestaukselle ovat aikataulujen sovittaminen asennuksille ja testaukselle, järjestelmien eriaikaisista kehityssykleistä seuraavat saatavuusongelmat ja näistä saatavuusongelmista tiedottaminen riittävän ajoissa. Vaikka järjestelmien kehityssyklit on pyritty synkronoimaan, on kehitystöiden asennuksista aiheutuvissa katkoissa välillä useamman päivän ero, jolloin kaikkien järjestelmien palvelut eivät olekaan saatavilla suunniteltuna integraatiotestauksen ajankohtana. Testijoukosta riippuen joudutaan koko integraatiotestaus tai osa siitä siirtämään toiseen ajankohtaan. Testaajille ja kaikille testaukseen osallistuville sopivan uuden ajankohdan löytäminen on sitä vaikeampaa, mitä myöhemmin tieto katkosta saadaan eli mitä lähempänä alun perin suunniteltua testausajankohtaa ollaan.

5.4 Testauksen prioriteetit ja muut vaatimukset

Eduskunnan lainsäädäntötyön tietojärjestelmien häiriöttömän toiminnan kannalta kriittisten

integraatioiden toiminta on integraatiotestauksessa aina varmistettava. Kriittisiä integraatioita ovat kaikki ydinjärjestelmien väliset integraatiot, joissa sekä palvelua tarjoava että palvelua käyttävä järjestelmä on jokin eduskunnan ydinjärjestelmistä. Integraatio luokitellaan kriittiseksi silloin, kun tällaisen integraation toimimattomuus aiheuttaa vakavia häiriöitä palvelua käyttävän ja tarvitsevan ydinjärjestelmän toimintaan, esimerkiksi mikäli täysistunnon hallintajärjestelmä ei saa täysistunnon päiväjärjestystä valtiopäiväasioiden ja –asiakirjojen hallintajärjestelmästä, ei täysistunnon hallintajärjestelmässä voida käynnistää täysistunnon hallintatoimia ilman erillisiä poikkeustoimia, eikä täysistunnon hallintajärjestelmä voi puolestaan tarjota palveluaan muille ydinjärjestelmille. Kriittiset ja integraatiotestauksessa korkeimmalla prioriteetilla testattavat integraatiot on kuvattu kuvassa 15.



Kuva 15. Korkeimman prioriteetin integraatiot eduskunnan tietojärjestelmien integraatiotestauksessa.

Kuvassa 15 on punaisella katkoviivalla kuvattu kaikki ne integraatiot tietosisältöineen, jotka testataan integraatiotestauksessa korkeimmalla prioriteetilla. Vaikka kamerarobotiikka ja verkkolähetys eivät ole eduskunnan ydinjärjestelmiä, ovat näille palveluja välittävät integraatiot kriittisiä palvelujen merkityksen vuoksi täysistunnon seurattavuudelle. Täysistuntoja on kaikkien mahdollista seurata verkkolähetyksen sekä YLE:n televisiolähetysten kautta.

Korkeimman prioriteetin integraatioista valtiopäiväasioiden ja –asiakirjojen hallintajärjestelmän ja täysistunnon hallintajärjestelmän integraatioiden testaus vie eniten resursseja vaatien vähintään 5 testaajan työtä ja aikaa yli 4 tuntia, kun lasketaan testauksen suunnitteluun, koordinointiin, suoritukseen ja raportointiin kuluva aika yhteensä. Hitaimmat testattavat ovat edustaja- ja toimielintietojen hallintajärjestelmän integraatiot, sillä sekä niiden kautta tarjottavien palvelujen muodostaminen edustaja- ja toimielintietojen hallintajärjestelmässä että käsitteleminen palveluja hyödyntävissä ydinjärjestelmissä on hidasta.

Integraatiotestauksen tuloksien seurannassa vuosien 2017 – 2019 aikana virhealtteimmiksi integraatioiksi nousevat edustaja- ja toimielintietojen hallintajärjestelmän integraatiot sekä valtiopäiväasioiden ja –asiakirjojen hallintajärjestelmän integraatio sähköpostiin. Edustaja- ja toimielintietojen hallintajärjestelmän integraatioiden testauksessa esiintyneet virheet eivät ole johtuneet integraatioiden virheellisestä toiminnasta, vaan mm. järjestelmien hitaudesta muodostaa, käsitellä ja päivittää oma tietosisältönsä näiden palveluiden pohjalta. Mikäli palvelua käyttävä järjestelmä esimerkiksi hyödyntää välimuistiin pohjautuvaa ratkaisua edustaja- ja toimielintietojen käytössä, vaatii edustajatietojen muutosten päivittäminen välimuistin tyhjennyksen tai virkistämisen.

Valtiopäiväasioiden ja –asiakirjojen hallintajärjestelmän integraatio sähköpostiin käsittää mm. valiokuntien kokouskutsun asiantuntijoille. Tässä integraatiossa esiintyneet virheet ovat johtuneet mm. merkistöstandardin ja liitteiden virheellisestä käsittelystä integraatiojärjestelmässä, jolloin kokouskutsu testiympäristössä ei ole saapunut sähköpostiin ollenkaan tai testaajalle näkyvästä sisällöstä on puuttunut skandinaaviset kirjaimet (ä, ö, å) tai viestin liitteet. Vaikka tämä integraatio ei ole kriittinen, koska kutsun voi lähettää myös valtiopäiväasioiden ja –asiakirjojen hallintajärjestelmän ulkopuolelta, sisällytetään tämän integraation testaus aina integraatiotestauksen testijoukkoon virhealttiutensa vuoksi.

6 Tapaustutkimus - eduskunnan tietojärjestelmien integraatiotestauksen automatisointi

Eduskunnan tietojärjestelmien integraatiotestauksen tavoitteena on varmistaa integraatioiden vaatimusten mukainen toiminta, eduskunnan tietojärjestelmien vaatimusten mukainen toiminta integraatioiden kautta välitettävien palvelujen tuottamisessa ja hyödyntämisessä sekä eduskunnan järjestelmien järjestelmän mission toteutuminen. Eduskunnan tietojärjestelmien välinen integraatiotestaus on tarpeen saada nopeammaksi ja resurssienkäytöltään tehokkaammaksi. Integraatioiden testitapauksista tulee tunnistaa häiriöttömän lainsäädäntötyön kannalta kriittisimmät, ja nämä pitää pystyä testaamaan tunnissa korkeintaan yhden testaajan toimesta. Kaikki integraatiot tulee pystyä testaamaan neljässä tunnissa. Integraatiotestauksen tehostamiseksi on testauksessa hyödynnettävä testausautomaatiota. Järjestelmien välisen integraatiotestauksen automatisoinnissa on huomioitava eduskunnan tietojärjestelmille ja toimintaympäristölle sopivat keinot ja työkalut. Integraatiotestauksen automatisoinnin tavoitteena on tehostaa eduskunnan integraatiotestausta sekä varmistaa integraatiotestauksen laatu ja riittävä kattavuus kaikissa testaustilanteissa.

6.1 Automatisoinnin kohteet

Automatisoinnin ensisijaisia kohteita ovat paljon aikaa ja testaajia vaativat integraatiotestit sekä kriittisten integraatioiden testit. Jatkuva integraatio ja testauksen hallinta ovat tärkeitä automatisoinnin kohteita muutosten asennusten, testauksen suunnittelun, testauksen organisoinnin ja testauksen raportoinnin tehostamisessa.

Paljon aikaa vievät integraatiotestit

Integraatiotestauksessa eniten aikaa vievät edustaja- ja toimielintietojen hallintajärjestelmän integraatioiden testaus. Osa edustaja- ja toimielintietojen hallintajärjestelmän integraatioiden kautta välitettävistä palveluista, kuten tilastotiedot, muodostetaan järjestelmässä ilman järjestelmän käyttäjän toimenpiteitä. Nämä palvelut muodostetaan ainoastaan yöaikaan. Tästä syystä erityisesti tilastotietojen julkaisun testaus on hidasta, sillä testaustuloksia on odotettava seuraavaan päivään. Myös käyttöliittymän kautta käynnistettyjen yksittäisten edustaja- ja toimielintietojen julkaisujen testaus on yksi eniten aikaa vievistä testitapauksista, sillä testauksen lopputulosta joudutaan odottamaan vähintään 40 minuuttia. Lopputuloksen odotuksen aikana voidaan kuitenkin testata muita integraatioita, joista eniten aikaa ja testaajia vaativat kriittisten integraatioiden testaus.

Kriittiset korkeimman prioriteetin integraatiotestit

Kriittisten integraatioiden testaus on automatisoitava, jotta nämä integraatiot voidaan testata riittävän tehokkaasti ja luotettavasti. Kriittisten integraatioiden manuaalinen testaus vie paljon

aikaa ja vaatii usean testaajan yhtäaikaista työtä. Kriittiset korkeimman prioriteetin integraatiot eduskunnan tietojärjestelmien integraatiotestauksessa on esitetty kuvassa 15. Integraatiotestauksen tarve nousee yksittäiseen järjestelmään tehtyjen muutosten, palvelinympäristöön tehtyjen muutosten tai tietoliikenneympäristöön tehtyjen muutosten jälkeen. Tällaisia muutoksia ovat esimerkiksi kehitystöiden asennus, versio- ja tietoturvapäivitysten asennus ja integraatioihin vaikuttavat konfiguraatiomuutokset. Yksittäiseen järjestelmään tehtyjen muutosten jälkeen integraatiotestauksessa keskitytään testaamaan erityisesti niitä integraatioita, joiden kautta muutoksen kohteena oleva järjestelmä tarjoaa palvelua muille järjestelmille tai käyttää muiden järjestelmien tarjoamia palveluja. Tietoliikenneympäristöön, palvelinympäristöön tai integraatiojärjestelmään tehtyjen muutosten jälkeen kaikki integraatiot on testattava, sillä muutokset voivat vaikuttaa kaikkiin integraatioihin. Ainoastaan integraatioiden testaus päästä päähän palveluja tarjoavista järjestelmistä palveluja käyttäviin järjestelmiin testaa sekä yksittäisten järjestelmien että palvelinympäristön ja tietoliikenneympäristön integraatioihin liittyvän toiminnan. Integraatiotestauksen automatisointi on riittävän testauskattavuuden ja luotettavuuden takaamiseksi toteutettava siten, että se testaa integraatiot päästä päähän.

Integraatiotestitapaukset on johdettu avainskenaarioista. Kriittisten integraatioiden testaus vaatii kaikkien niiden avainskenaarioiden sisältämien käyttötapausten suorittamisen, joissa kriittisiä integraatioita käytetään. Kuten avainskenaarioiden käyttötapaukset niin myös näiden testitapaukset on suoritettava tietyssä järjestyksessä tiettyyn aikaan, jotta kaikki kriittiset integraatiot voidaan testata. Testitapausten ja testaustehtävien järjestys ja ajoitus on huomioitava myös kriittisten integraatioiden testauksen automatisoinnissa. Yksittäisissä järjestelmissä suoritettavia integraatiotestauksessa tarvittavia testaustehtäviä ei voida automatisoinnissa jättää suorittamatta tai korvata jäljitelmillä. Esimerkiksi jättämällä suorittamatta valtiopäiväasioiden ja –asiakirjojen järjestelmässä täysistunnon ja sen päiväjärjestyksen laatiminen korvaamalla täysistunnon hallintajärjestelmään lähetettävä päiväjärjestys jäljitelmällä, ei saada testattua kaikkia kriittisiä integraatioita. Tällöin saadaan testattua integraatiojärjestelmän ja täysistunnon hallintajärjestelmän välinen integraatio, mutta ei voida testata täysistunnon hallintajärjestelmän ja valtiopäiväasioiden ja –asiakirjojen hallintajärjestelmän välistä kriittistä integraatiota. Täysistunnon hallintajärjestelmästä ei voida tällöin lähettää esim. täysistunnon kulkutietoja ja puheenvuorotietoja, koska valtiopäiväasioiden ja –asiakirjojen hallintajärjestelmässä ei ole kyseistä täysistuntoa, joka ottaisi vastaan tietoja täysistunnon hallintajärjestelmältä.

Kriittisten integraatioiden testauksen automatisointi on järkevää toteuttaa automatisoimalla ensin palveluita tuottavien ydinjärjestelmien avainskenaarioiden mukaiset käyttötapaukset ja jatkokäyttämällä näitä automatisoituja testejä integraatiotestauksen automatisoinnissa. Tällä tavoin voidaan varmistaa automatisoitujen integraatiotestien oikeanlainen ja vaatimusten mukainen vuorovaikutus ydinjärjestelmien välillä ja ydinjärjestelmien ympäristön kanssa. Lisäksi näin

saadaan integraatiotestauksen automatisoinnin lopputuloksena järjestelmien manuaalisen testauksen tueksi validia testiaineistoa ja näin tehostettua myös mahdollisesti jäljelle jäävää manuaalista testausta. Avainskenaarioiden mukaiset käyttötapaukset kriittisten integraatioiden testauksessa on esitetty taulukossa 3.

Ydinjärjestelmä	**Käyttötapaus**
Valtiopäiväasioiden ja -asiakirjojen hallintajärjestelmä	Laadi päiväjärjestys. Edellytyksenä, että valtiopäiväasia, joka lisätään päiväjärjestykseen on tätä ennen luotu järjestelmään, valtiopäiväasialla on määrätyt tiedot ja se on oikeassa prosessinaskeleessa.
Valtiopäiväasioiden ja -asiakirjojen hallintajärjestelmä	Lähetä päiväjärjestys ja siirrä täysistunto seuraavaan vaiheeseen
Täysistunnon hallintajärjestelmä	Käynnistä täysistunto
Täysistunnon hallintajärjestelmä	Siirry seuraavaan asiakohtaan (sisältää asiakohdan päätöstietojen tallennuksen). Tämä suoritetaan jokaiselle täysistunnon päiväjärjestyksen mukaiselle asiakohdalle, paitsi viimeiselle.
Täysistunnon hallintajärjestelmä	Lisää puheenvuoro
Täysistunnon hallintajärjestelmä	Lisää äänestys
Täysistunnon hallintajärjestelmä	Viimeisen asiakohdan päätöstietojen tallennus
Täysistunnon hallintajärjestelmä	Lopeta täysistunto

Taulukko 3. Avainskenaarioiden käyttötapaukset ydinjärjestelmissä korkeimman prioriteetin integraatioiden testaamiseksi

Integraatiotestausautomaation tulee suorittaa taulukossa 3 listattujen käyttötapausten mukaiset testitapaukset oikeassa järjestyksessä oikeaan aikaan, jotta kaikki kriittiset integraatiot tulevat testattua. Taulukossa on ensimmäisessä sarakkeessa ydinjärjestelmä, jossa samalla rivillä

toisessa sarakkeessa oleva käyttötapaus tulisi testauksessa suorittaa. Testausautomaation tulee ensimmäiseksi luoda valtiopäiväasioiden ja -asiakirjojen hallintajärjestelmässä uusi Hallituksen esitys -tyyppinen valtiopäiväasia, syöttää valtiopäiväasialle järjestelmän vaatimat tiedot sekä lisätä valtiopäiväasialle vireilletuloasiakirja -tyyppinen valtiopäiväasiakirja. Tämän jälkeen testausautomaation tulee siirtää valtiopäiväasia järjestelmässä oikeaan prosessinaskeleeseen. Nämä tehtävät ovat edellytyksenä taulukon 3 ensimmäisen rivin käyttötapauksen suorittamiselle, josta kriittisten integraatioiden testaus alkaa. Testausautomaation tulee laatia täysistunnon päiväjärjestys, johon se valitsee aiemmin luomansa valtiopäiväasian. Tämän jälkeen testausautomaatio lähettää päiväjärjestyksen integraatiojärjestelmään muita ydinjärjestelmiä palvelemaan ja siirtää täysistunnon valtiopäiväasioiden ja -asiakirjojen hallintajärjestelmässä vaiheeseen, jossa järjestelmä voi mm. lukea täysistunnon hallintajärjestelmän lähettämiä täysistunnon kulkutietoja.

Täysistunnon hallintajärjestelmä ja päättyneiden täysistuntojen hallintajärjestelmä lukevat valtiopäiväasioiden ja -asiakirjojen hallintajärjestelmän lähettämän päiväjärjestyksen ja luovat sen pohjalta uuden täysistunnon. Täysistunnon hallintajärjestelmä lähettää lisäksi sähköpostikuittauksen päiväjärjestyksen vastaanottamisesta sen testiympäristöön määritetylle jakeluryhmälle. Julkaisujen hallintajärjestelmä lukee myös valtiopäiväasioiden ja -asiakirjojen hallintajärjestelmän lähettämän päiväjärjestyksen ja luo sen pohjalta täysistunto-verkkosivun ja aktivoi toiminnallisuuden, jonka kautta voidaan lisätä puheenvuorovarauksia ko. täysistuntoon. Nämä kaikki toiminnot tapahtuvat automaattisesti täysistunnon-, päättyneiden täysistuntojen- ja julkaisujen hallintajärjestelmässä aina, kun uusi päiväjärjestys on luettavissa, joten näitä tapahtumia varten ei tarvitse toteuttaa erillistä testausautomaatiota.

Testausautomaation tulee seuraavaksi käynnistää täysistunto täysistunnon hallintajärjestelmässä, josta täysistunnon hallintajärjestelmä lähettää tiedon integraatiojärjestelmään muita järjestelmiä varten. Valtiopäiväasioiden ja -asiakirjojen hallintajärjestelmä, äänitallenteiden hallintajärjestelmä, julkaisujen hallintajärjestelmä sekä täysistunnon hallintajärjestelmään integroidut eduskunnan ulkopuoliset järjestelmät lukevat tiedon täysistunnon käynnistämisestä ja suorittavat toimintoja luetun tiedon pohjalta, esimerkiksi valtiopäiväasioiden ja -asiakirjojen hallintajärjestelmä päivittää tietyn komponentin käyttöliittymään täysistunnon alkuajan ja julkaisujen hallintajärjestelmä päivittää Täysistunto-verkkosivulle tiedon täysistunnon käynnistymisestä. Täysistunnon hallintajärjestelmän käynnistymistietojen lukemisesta seuraavat toiminnot tapahtuvat järjestelmissä automaattisesti eikä testausautomaation tarvitse suorittaa näitä toimintoja. Järjestelmät pystyvät tämän jälkeen lukemaan täysistunnon hallintajärjestelmän seuraavaksi lähettämiä täysistunnon kulkutietoja, kuten nimenhuudon tulostiedot, asiakohdissa pidettyjen puheenvuorojen tiedot ja äänestystiedot.

Täysistunnon käynnistämisen jälkeen testausautomaation tulee siirtää täysistunto täysistunnon

hallintajärjestelmässä päiväjärjestyksen mukaiseen ensimmäiseen asiakohtaan, lisätä päätöstiedot asiakohtaan, siirtyä päiväjärjestyksen mukaiseen toiseen asiakohtaan, lisätä päätöstiedot toiseen asiakohtaan, siirtyä päiväjärjestyksen mukaiseen kolmanteen asiakohtaan, lisätä puheenvuorotiedot, lisätä äänestystiedot, lisätä päätöstiedot kolmanteen asiakohtaan ja siirtyä päiväjärjestyksen viimeiseen asiakohtaan. Jokaisesta täysistunnon kulkuun liittyvästä tiedosta, kuten päätöstieto, puhujatieto ja siirtyminen asiakohtaan lähtee täysistunnon hallintajärjestelmästä tieto integraatiojärjestelmään muiden järjestelmien käytettäväksi. Jokainen täysistunnon kulkutietoa lukeva järjestelmä automaattisesti päivittää tietonsa lukemiensa täysistunnon kulkutietojen pohjalta, esimerkiksi julkaisujen hallintajärjestelmä päivittää Täysistunto-verkkosivulle tiedon siitä, mikä päiväjärjestyksen mukainen asiakohta on parhaillaan käsittelyssä ja kuka parhaillaan puhuu täysistunnossa. Päättyneiden täysistuntojen hallintajärjestelmä automaattisesti laskee äänestystietojen tuloksen täysistunnon hallintajärjestelmän lähettämien äänestystietojen pohjalta ja lähettää äänestystulokset integraatiojärjestelmään. Muut järjestelmät automaattisesti lukevat päättyneiden täysistuntojen hallintajärjestelmän lähettämät äänestystulokset ja päivittävät omaa sisältöään näiden pohjalta, esim. julkaisujen hallintajärjestelmä muodostaa äänestystuloksista oman verkkosivun ja lisää Täysistunto-verkkosivulle päiväjärjestyksen mukaisen asiakohdan alle linkin tälle äänestystulossivulle.

Siirrettyään täysistunnon päiväjärjestyksen mukaiseen viimeiseen asiakohtaan testausautomaation tulee lisätä viimeisen asiakohdan päätöstiedot ja päättää eli lopettaa täysistunto täysistunnon hallintajärjestelmässä. Täysistunnon hallintajärjestelmä automaattisesti lähettää täysistunnon lopetuksesta tiedon integraatiojärjestelmään. Muut järjestelmät automaattisesti lukevat täysistunnon lopetustiedot ja päivittävät sisältönsä sekä tilansa tämän mukaan, esim. valtiopäiväasioiden ja -asiakirjojen hallintajärjestelmä päivittää tietyn komponenttinsa käyttöliittymälle täysistunnon päättymisajan. Kriittisten integraatioiden testaus ja testausautomaatio päättyy tähän.

Automatisoimalla taulukossa listattujen käyttötapausten testitapaukset ja ketjuttamalla nämä yhteen saadaan kaikkien kriittisten integraatioiden testitapaukset automatisoitua lukuun ottamatta testitapausten lopputulosten tarkistusta. Integraatiotestauksen keskeytysehtojen tarkistamista ja virheiden nopeaa havaitsemista varten tulee testausautomaation tarkistaa kunkin testitapauksen lopputulokset ko. testitapauksen suorituksen lopuksi, esim. valtiopäiväasioiden ja -asiakirjojen hallintajärjestelmän lähetettyä päiväjärjestys integraatiojärjestelmään, tulee testausautomaation tarkistaa lähetyksen onnistuminen integraatiojärjestelmään. Mikäli päiväjärjestyksen lähetys ei onnistunut, testausautomaation tulee keskeyttää testauksen suoritus ja ilmoittaa mahdollisesta virheestä. Testausautomaation tulee lisäksi tarkistaa täysistunnon kulkutietojen päivittyminen julkaisujen hallintajärjestelmän Täysistunto-sivulle, valtiopäiväasioiden ja -asiakirjojen hallintajärjestelmään sekä äänitallenteiden hallintajärjestelmään. Ilman täysis-

tunnon kulkutietojen automaattista tarkistusta järjestelmien käyttöliittymistä, testausautomaation lopputulosten todentaminen järjestelmien käyttöliittymistä käsin jäisi eduskunnan testaajien tehtäväksi.

Loput integraatiotestit, jotka testaavat luvussa 4.2 kuvissa 10 ja 12 esitettyjen avainskenaarioiden mukaiset integraatiot automatisoidaan paljon resursseja vievien ja kriittisten integraatioiden testauksen jälkeen. Kuvissa 10 ja 12 esitettyjen valiokunnan kokouskutsun lähettämisen ja EU-asioiden käsittelytietojen jakamisen manuaalinen testaus vie yhdeltä testaajalta noin 20 minuuttia. Nämä integraatiotestit voidaan suorittaa rinnakkain samaan aikaan kriittisten integraatioiden testauksen kanssa.

Jatkuva integraatio

Integraatioiden automatisoidut testit voidaan suorittaa ajastetusti tai aina tarvittaessa hyödyntämällä jatkuvaa integraatiota (Continuous Integration, CI). Jatkuvan integraation avulla automatisoidut testit saadaan pidettyä paremmin ajan tasalla, sillä tällöin testejä käytetään jatkuvasti ja testien päivitystarpeista saadaan nopeasti tieto. Yksittäisten järjestelmien automatisoitujen testien, joissa ei testata integraatioita tai integraatioiden tilalla on käytetty jäljitelmiä, ajaminen ei vaadi muiden ydinjärjestelmien saatavuutta. Nämä testit voidaan ajaa ajastetusti päivittäin. Mikäli halutaan testata järjestelmien integraatioiden toimintaa, on kaikkien ydinjärjestelmien oltava saatavissa, sillä muuten testit epäonnistuvat. Järjestelmien ajoittain eriaikaisten kehityssyklien vuoksi järjestelmien saatavuus vaihtelee. Järjestelmien saatavuus tulee varmistaa ennen automatisoitujen integraatiotestien ajamista. Testausautomaatioon tulee liittää testausehtojen eli järjestelmien saatavuuden tarkistus. Tämän jälkeen integraatiotestit voidaan ajastaa jatkuvan integraation avulla. Testien ajan tasalla pysymisen lisäksi muita jatkuvan integraation hyötyjä ovat mahdollisuus ajaa automatisoituja testejä rinnakkain sekä järjestelmien välisten integraatioiden toiminnasta ja integraatiojärjestelmän laadusta nopeasti saatava tieto. Automatisoituja integraatiotestejä ei kuitenkaan voida ajaa samaan aikaan yksittäisten järjestelmien omien, integraatioiden jäljitelmiä hyödyntävien testien kanssa, sillä esim. täysistunnon hallintajärjestelmässä ei voi olla kahta samanaikaista täysistuntoja eikä valtiopäiväasioiden ja –asiakirjojen hallintajärjestelmä voi ottaa samanaikaisesti vastaan tietoja kahta eri täysistuntoa koskien.

Suunnittelu ja raportointi

Testausautomaatiotyökalujen käyttö tehokkaampaan testitapausten suunnitteluun ja testauksen raportointiin säästää testauksen kustannuksia helposti enemmän kuin testauksien suorituksen automatisointi (Berner, Weber, &Keller, 2005, s.573). Integraatiotestauksen suunnittelun ja raportoinnin apuna käytetään testauksen hallinnan työkalua, JIRA-järjestelmää, jonne kaikki integraatiotestitapaukset on kirjattu. Integraatiotestauksen käytössä oleva JIRA-järjestelmä ei kuitenkaan tue testaussuunnitelman ja testijoukon hallintaa eikä sinne voi tuoda automatisoituja testejä. Myös testaustuloksien raportointi on käytössä olevassa JIRA-järjestelmässä riittämätön

testijoukon hallinnan puutteen vuoksi sekä siksi, että testitapauksien suorituksen tuloksia ei voi raportoida testaustehtävä tai testiaskel kerrallaan. Testauksen hallinnan työkalun avulla vaatimukset saadaan tuotua testitapauksien suunnitteluun ja jokaisen vaatimuksen testauksen tilannetta voidaan seurata ja jäljittää (Spillner, Linz, & Schaefer, 2014, s. 286). Käytössä oleva JIRA-järjestelmä ei mahdollista vaatimusten yhdistämistä testitapauksiin siten, että niiden testausta voitaisiin seurata ja raportoida.

JIRA-järjestelmän käyttöä integraatiotestauksen suunnittelussa, suorituksessa ja raportoinnissa tulee tehostaa. Lisäosin täydennetystä JIRA-järjestelmästä on saatu merkittäviä hyötyjä valtiopäiväasioiden ja –asiakirjojen hallintajärjestelmän testauksessa, ja samat lisäosat ja käytännöt tulisi ottaa käyttöön myös integraatiotestauksen JIRA-järjestelmässä.

6.2 Kontekstin merkitys automatisoinnissa

Testaajalle näkymätön testausautomaatio on yksi yleisistä testauksen automatisoinnin ongelmista (Berner, Weber, &Keller, 2005, s.577). Eduskunnan testaajille yksi testausautomaation tärkeistä tehtävistä on tuottaa testiaineistoa mahdollista manuaalista jatkotestausta varten. Manuaalinen jatkotestaus on tarpeen usein silloin, kun integraatiotestauksessa varmistetaan jonkin tietyn ydinjärjestelmän toiminta muutosten jälkeen, esimerkiksi valtiopäiväasioiden ja –asiakirjojen hallintajärjestelmässä halutaan sinne tehtyjen muutosten jälkeen testata päättyneiden täysistuntojen hallintajärjestelmän lähettämien tietojen nouto valtiopäiväasiakirjoille. Tällöin testausautomaation lopputulosten tulee olla nähtävissä järjestelmien käyttöliittymistä, jotta testaaja tietää, minkä täysistunnon tiedot ovat testiympäristössä hyödynnettävissä jatkotestaukseen. Testausautomaation suorittaman integraatiotestauksen tulosten tarkastelu käyttöliittymistä ei saa olla pakollista, vaan testausautomaation tulee tuottaa helposti ymmärrettävät tarkat raportit testauksen tuloksista.

Eduskunnan testiympäristö on tuotantoympäristön kaltainen koskien kaikkia järjestelmiä, palvelinympäristöä ja tietoliikenneympäristöä. Vaikka testiaineisto ei vastaa tuotantoaineistoa, koskee testiaineiston luontia samat määritykset kuin tuotantoaineiston, esim. valtiopäivävuosirajauksia koskien. Valtiopäiväasioiden, -asiakirjojen ja täysistuntojen tunnisteen numero-osa noudattaa juoksevaa numerointia, esimerkiksi eduskunnan verkkosivuilla näkyvä PJ 87/2019 vp tarkoittaa valtiopäivävuoden 2019 87. täysistunnon päiväjärjestystä. Kriittisten integraatioiden testauksen edellytyksenä on, että testiaineiston tunnisteiden numerointi on kaikissa ydinjärjestelmissä yhteneväinen. Mikäli testiympäristön täysistunnon hallintajärjestelmässä täysistunnon tunnisteen numero-osa on eri kuin valtiopäiväasioiden ja –asiakirjojen hallintajärjestelmässä, ei näiden järjestelmien välisten integraatioiden kautta ole mahdollista välittää ja kohdistaa palveluja oikein.

Integraatiotestauksessa tuotettua aineistoa ei ole mahdollista poistaa testiympäristöstä ilman,

että poistot tehdään jokaiseen järjestelmään. Tämän lisäksi jokaisen järjestelmän valtiopäiväasioiden, -asiakirjojen, kokousten ja täysistuntojen numeroinnit tulisi palauttaa testausta edeltävään tilaan. Mikäli yksikin poisto tai palautus epäonnistuu tai unohtuu tehdä, eivät seuraavan kerran suoritettavat integraatiotestit onnistu ongelmitta. Tämän vuoksi testausautomaatio ei saa tuottaa integraatiotestauksessa sellaista aineistoa testiympäristöön, joka pitäisi testauksen päätyttyä poistaa.

6.3 Oletusten ja reunaehtojen haastaminen

Eduskunnan ydinjärjestelmät ovat toiminnallisesti ja hallinnollisesti itsenäisiä järjestelmiä. Järjestelmien hallinta, kehitys, testaus sekä ylläpito on järjestetty järjestelmäkohtaisesti. Järjestelmien itsenäisyys tuo haasteita järjestelmien integraatiotestaukseen, jossa vastuut testauksen organisoinnista, suorituksesta sekä oikean tiedon välittymisestä oikealle järjestelmälle tulee olla selvät. Järjestelmien luotettava ja kattava integraatiotestaus on mahdollista tehdä ainoastaan eduskunnan testiympäristössä, jossa kaikki ydinjärjestelmät ovat saatavilla ja integraatioiden tietoliikenneympäristö on tuotantoympäristön kaltainen.

Ydinjärjestelmien itsenäisyys ei kuitenkaan ole este järjestelmien välisten integraatioiden osittaiselle testaamiselle ydinjärjestelmien omissa kehitysympäristöissä. Yksittäisten ydinjärjestelmien omassa järjestelmätestauksessa ydinjärjestelmän integraatiot muihin ydinjärjestelmiin on mahdollista testata mm. jäljitelmiä hyödyntäen. Tällöin testaus voidaan suorittaa muussakin kuin eduskunnan testiympäristössä. Esimerkiksi valtiopäiväasioiden ja –asiakirjojen hallintajärjestelmän toimittaja testaa järjestelmätesteissään täysistunnon aikana käytettävissä olevia toimintoja ja täysistunnon aikana laadittavia asiakirjoja hyödyntämällä SoapUI:lla toteutettua jäljitelmää täysistunnon hallintajärjestelmän integraatiosta. Jäljitelmä jäljittelee täysistunnon hallintajärjestelmän lähettämää täysistunnon käynnistymisen sanomaa, jolloin valtiopäiväasioiden ja -asiakirjojen hallintajärjestelmä aktivoi täysistunnon aikana käytettävät toimintonsa. Tämän kaltaisella järjestelmätestauksella integraatioiden toiminta järjestelmissä saadaan hyvin pitkälle varmistettua ja automatisoitua ennen järjestelmien integraatiotestausta eduskunnan testiympäristössä. Eduskunnan järjestelmien välisten integraatioiden toiminnan testaaminen eduskunnan testiympäristössä vaatii jokaisen ydinjärjestelmän testausta integraatioihin liittyvien käyttötapausten osalta. Kun integraatiotestaus on jäljitelmien avulla testattu ja automatisoitu yksittäisissä järjestelmissä, voidaan näitä automatisoituja testejä hyödyntää järjestelmien integraatiotestauksen automatisoinnissa.

Kaikkien integraatioiden testauksessa tulee kaikkien järjestelmien testiympäristöjen olla saatavissa eduskunnan testiympäristössä palveluiden voimakkaiden riippuvuuksien vuoksi. Tästä reunaehdosta voidaan kuitenkin joustaa tiettyjen järjestelmien osalta korvaamalla nämä jäljitel-

millä myös eduskunnan testiympäristössä tapahtuvassa integraatiotestauksessa. Valtiopäiväasioiden ja –asiakirjojen hallintajärjestelmän, täysistunnon hallintajärjestelmän sekä päättyneiden täysistuntojen hallintajärjestelmän tulee olla aina saatavilla integraatiotestauksessa, mutta edustaja- ja toimielintietojen hallintajärjestelmän sekä julkaisujen hallintajärjestelmän integraatiot on mahdollista korvata jäljitelmillä silloin, kun testaustarpeet eivät kohdistu nimenomaan näiden järjestelmien integraatioiden toiminnan varmistamiseen. Edustaja- ja toimielintietojen hallintajärjestelmän tarjoamat palvelut, yhtä lukuun ottamatta, eivät riipu muiden järjestelmien palveluista. Edustaja- ja toimielintietojen hallintajärjestelmän palveluiden jäljitelmät voivat tarjota palvelun muille järjestelmille samanlaisena kerta toisensa jälkeen. Ainoa edustaja- ja toimielintietojen hallintajärjestelmän toisen järjestelmän palvelusta riippuva palvelu on täysistunnon hallintajärjestelmälle tarjottava palvelu, joka riippuu täysistunnon hallintajärjestelmän pyyntöpalvelusta. Tämäkin voidaan tarjota jäljitelmän avulla, sillä palvelun sisältö ei ole riippuvainen pyyntöpalvelusta.

Jäljitelmää on mahdollista käyttää myös julkaisujen hallintajärjestelmän ja täysistunnon hallintajärjestelmän välisen integraation testauksessa, jolloin täysistunnon hallintajärjestelmä saakin puheenvuorovarauksen jäljitelmältä eikä julkaisujen hallintajärjestelmältä. Jäljitelmä ei tässä tapauksessa voi tarjota palvelua aina samanlaisena, vaan jäljitelmän tarjoaman palvelun sisältö tulee joka kerran muokata integraatiotestauksessa käytettävän täysistunnon tunnistetiedoilla. Palvelun yksinkertaisuuden ja pienen tietosisällön vuoksi sisällön generointi jäljitelmään on nopeaa.

Integraatiotestauksessa käytettävät jäljitelmät tulee lähettää integraatiojärjestelmästä. Jäljitelmiä käyttämällä integraatioista jää näin ollen testaamatta vain jäljiteltävää palvelua tarjoavan järjestelmän ja integraatiojärjestelmän välinen integraatio. Jäljitelmiä voi käyttää ainoastaan niissä tapauksissa, kun edustaja- ja toimielintietojen hallintajärjestelmä tai julkaisujen hallintajärjestelmä eivät ole saatavissa integraatiotestaukseen. Jäljitelmiä ei voi käyttää silloin, kun testauksen kohteena on kaikki integraatiot tietoliikenneympäristöön, palvelinympäristöön tai integraatiojärjestelmään tehtyjen muutosten jälkeen.

7 Automatisoinnin toteutusvaihtoehtojen evaluointi

Eduskunnan integraatiotestauksen automatisoinnin ratkaisuissa tulee huomioida järjestelmien itsenäisyydestä aiheutuvat haasteet integraatiotestaukselle. Koska integraatiot tulee testata päästä päähän avainskenaarioiden mukaisesti ja itsenäisten järjestelmien järjestelmätason eli sovelluslogiikan koodi ei ole järjestelmien järjestelmän saatavilla, on integraatiotestauksen automatisoinnin ratkaisun perustuttava järjestelmien käyttöliittymien kautta saavutettavaan koodiin. Automatisoinnin ratkaisussa vastuut testauksen organisoinnista, suorituksesta ja automaattisten testien ylläpidosta on oltava selvät ja sovitut.

Integraatiotestauksen automatisoinnin ratkaisua arvioitaessa tulee kiinnittää huomiota myös automatisoinnin kustannuksiin. Käyttöliittymien kautta nauhoitetut testiskriptit ovat herkkiä rikkoutumaan pienimmästäkin käyttöliittymää koskevasta muutoksesta (Thummalapenta, Sinha, Singhania, & Chandra, 2012, s. 881). Käyttöliittymiin perustuva testausautomaatio tukeutuukin usein skriptipohjaiseen testaukseen (Hu, Zhu & Yang, 2018, s. 269). Visuaalisen käyttöliittymän kautta testaukseen liittyy useita haasteita kuten testiskriptien pitkä suoritusaika, korkeat ylläpitokustannukset ja korkeat virheellisten positiivisten virhehavaintojen analysointiin kuluvat kustannukset (Alégroth, Karlsson & Radway, 2018, s. 180-181). Käyttöliittymiin perustuvien testien ylläpidon kustannuksiin voidaan vaikuttaa testiskriptien modulaarisella arkkitehtuurilla, joka mahdollistaa testiskriptin osien itsenäisen ylläpidon erillään skriptin muista osista (Alégroth, Feldt & Kolström, 2016, s.76). Avainsanapohjainen testi koostuu avainsanoista, jotka ovat viittauksia tiettyyn joukkoon käyttöliittymässä suoritettavia toimintoja (ISO/IEC/IEEE International Standard, 2016, s.3). Avainsanapohjainen testi on modulaarinen ja säästää testien ylläpitokustannuksia, sillä käyttöliittymän tietyn toiminnallisuuden muututtua riittää ylläpitää vain tietyn avainsanan takana oleva toiminnallisuus, jolloin tämän asiasanan sisältäviin testeihin ei tarvitse tehdä mitään muutoksia (ISO/IEC/IEEE International Standard, 2016, s.5). Myös testiskriptien lineaarinen rakenne vähentää virheiden analysoinnin kompleksisuutta, jonka vuoksi silmukoita ja haaroja tulee skriptissä välttää tekemällä näistä omat itsenäiset skriptit (Alégroth, Feldt & Kolström, 2016, s.76). Integraatiotestauksen automatisoinnin ratkaisun tulee tukea testiskriptien modulaarista arkkitehtuuria ja useiden itsenäisten skriptien yhdistämistä eduskunnan järjestelmien integraatiot testaavaksi skriptiksi.

Käyttöliittymiin perustuvan testausautomaation suorituksessa havaitut virheet johtuvat usein testiskriptien rikkoutumisesta eikä virheistä testattavissa järjestelmissä (Hu, Zhu & Yang, 2018, s. 269). Testiskriptien rikkoutumisella tarkoitetaan tässä yhteydessä testiskriptien vanhenemista sen vuoksi, että järjestelmien käyttöliittymäkoodin muuttuessa ei ole vastaavasti päivitetty testiskriptejä (Hu, Zhu & Yang, 2018, s. 269 ja Pinto, Sinha & Orso, 2012, s. 1). Virheellisten positiivisten virhehavaintojen analysointiin kuluvan ajan säästämiseksi automatisoidun integraatiotestauksen tulee tuottaa lopputuloksista selkeä raportti sekä suorituksen aikana osoittaa

ja raportoida selkeästi mahdolliset virheet ja virheiden syyt testiskripteissä.

Tämän luvun alaluvuissa esitellään eduskunnan tietojärjestelmien integraatiotestauksen automatisoinnin tavoitetila, automatisoinnin vaihtoehtoiset ratkaisut sekä ratkaisujen evaluointi. Automatisoinnin vaihtoehtoisten ratkaisujen evaluointia varten eduskunnan integraatiotestauksen automatisointia toteutettiin käytännön kokeilussa kahdella eri automatisointityökalulla.

7.1 Automatisoinnin tavoitetila

Eduskunnan tietojärjestelmien integraatiotestauksen automatisoinnin tavoitetilassa kaikkien eduskunnan kriittisten korkeimman prioriteetin integraatiotestitapausten suoritus on automatisoitu. Tämä mahdollistaa integraatiotestien suorittamisen hyvin lyhyelläkin varoitusajalla esimerkiksi kriittisten tietoturvapäivitysten asennusten jälkeen. Korkeimman prioriteetin integraatiotestitapausten lisäksi kaikki usein toistettavat regressiotestitapaukset on automatisoitu, jolloin testausautomaatio kattaa kaikkien integraatioiden testauksen mukaanlukien kokouskutsujen lähetyksen sähköpostiin sekä EUTORI- ja IPEX-integraatiot.

Tavoitetilassa integraatiotestauksen suunnitteluun, organisointiin ja koordinointiin ei kulu aikaa. Automatisoitu integraatiotestaus suoritetaan joka kerran eduskunnan testiympäristössä, kun integraatioita on tarpeen testata. Integraatiotestiskriptit suorittavat tarvittavat tarkistukset testauksen jälkeen järjestelmien käyttöliittymistä, esim. eduskunnan verkkosivuilta, että kaikki integraatiotestauksessa julkaistu aineisto löytyy verkkosivuilta. Integraatiotestauksessa syntyvä testiaineisto on hyödynnettävissä integraatioiden tai järjestelmien manuaaliseen jatkotestaukseen. Integraatiotestiskriptien ajon aikainen suoritus on monitoroitavissa integraatiojärjestelmästä sekä lopputulokset ovat tarkistettavissa integraatiojärjestelmän lokeilta ja suorituskykyraportilta.

Integraatiotestiskriptien suoritus tapahtuu eduskunnan jatkuvan integraation Jenkins-palvelimella myös ajastetusti. Jenkins-palvelimella ajetaan ensin skripti, joka tarkistaa integraatiotestauksessa tarvittavien järjestelmien testiympäristöjen saatavuuden ja tämän onnistunut suoritus sekä järjestelmien testiympäristöjen saatavuus ovat edellytyksenä integraatiotestiskriptien suoritukselle. Jatkuvan integraation avulla integraatiotestiskriptit saadaan pidettyä ajan tasalla ja teknistä velkaa testien ylläpitoon liittyen ei kerry liian pitkältä ajalta. Integraatiotestiskriptit ovat avainsanapohjaisia ja modulaarisia koostuen useista itsenäisistä testiskripteistä, joita voidaan ylläpitää itsenäisesti. Automatisoitu integraatiotestaus tuottaa testauksen lopputuloksista raportin, josta on tulosten lisäksi virheiden sattuessa nähtävissä, missä testiskriptin askeleissa virheitä havaittiin ja miksi testiautomaatio tulkitsee nämä virheeksi.

7.2 Automatisoinnin vaihtoehtoiset ratkaisut

Käyttötapausten pohjalta laadittujen testitapausten automatisointia on toteutettu tähän mennessä täysistunnon hallintajärjestelmässä ja valtiopäiväasioiden ja –asiakirjojen hallintajärjestelmässä. Muissa ydinjärjestelmissä käyttötapausten mukaista testausta ei ole automatisoitu vaan testausautomaatio on kohdistunut mm. suorituskyky- ja kuormatestaukseen. Yhtenä automatisoinnin vaihtoehtoisena ratkaisuna on laajentaa täysistunnon hallintajärjestelmässä ja valtiopäiväasioiden ja –asiakirjojen hallintajärjestelmässä toteutettua testausautomaatiota koskemaan myös avainskenaarioiden mukaiset käyttötapaukset, joissa nämä järjestelmät käyttävät ja tarjoavat palveluja toisilleen ja muille järjestelmille integraatioidensa kautta. Täysistunnon hallintajärjestelmän ja valtiopäiväasioiden ja –asiakirjojen hallintajärjestelmän käyttöliittymät ovat selainpohjaisia. Käyttötapausten pohjalta laadittuja järjestelmien käyttöliittymiin perustuvia testitapauksia on automatisoitu Robot Framework automatisointikehyksen avulla. Robot Framework on modulaarinen testiautomaatiokehys avainsanapohjaisten testiskriptien laadintaan, joiden avulla luodaan testiaineistoa ja suoritetaan toimintoja testauksen kohteena olevassa järjestelmässä (Robot Framework, 2020).

Koska eduskunnan järjestelmien integraatiotestaus käsittää eri ydinjärjestelmien toimintojen suorittamista, vaatii avainsanoihin pohjautuvien kattavien integraatiotestiskriptien laatiminen ja testien lopputilan tarkistukset kaikkien integraatioihin osallistuvien järjestelmien käyttöliittymäkoodin hyvää tuntemusta. Yksittäisten ydinjärjestelmien kehittäjät tai ylläpitäjät eivät tunne toisten järjestelmien saatikka kaikkien integraatioihin osallistuvien järjestelmien käyttöliittymäkoodia, joten integraatiotestiskriptien laatiminen Robot Framework automatisointikehyksellä vaatii eri ydinjärjestelmien ylläpitäjien yhteistyötä.

Robot Framework automatisointikehyksen käyttö integraatiotestauksen automatisointiin edellyttää järjestelmien käyttöliittymien noudattavan elementtien yksilöivää nimeämiskäytäntöä. Jokaisella elementillä tulee olla uniikki tunniste, kuten ID tai nimi, jonka avulla elementti paikallistetaan testiskriptiä ajettaessa. Uniikkien tunnisteiden muuttumattomuuteen tulee panostaa esim. kommentoimalla koodiin tunnisteen olevan testiautomaation käyttämä. Uniikin tunnisteen muuttuessa joudutaan Robot Framework automatisointikehyksen avulla laadittujen testitapausten testiautomaatiota ylläpitämään.

Vaihtoehto Robot Framework automatisointikehykselle on automatisoida integraatiotestitapaukset jollain muulla selainpohjaisiin käyttöliittymiin perustuvalla testiautomatisointityökalulla, kuten Selenium WebDriver työkalulla. Selenium WebDriverilla toteutetut testiskriptit suorittavat toimintoja käyttöliittymien kautta hyvin samaan tapaan kuin Robot Framework automatisointikehyksellä toteutetut. Robot Framework automatisointikehykseen on myös mahdollista asentaa Selenium kirjasto (SeleniumLibrary), joka käyttää Selenium WebDriverin moduuleita

(Robot Framwork SeleniumLibrary, 2020). Eduskunnan tähän mennessä Robot Framework automatisoinitkehyksellä toteutetut automatisoidut testit käyttävät mm. Selenium kirjastoa. Koska itsenäisten järjestelmien testejä on toteutettu Robot Framework automatisointikehyksellä Selenium kirjastoa hyödyntäen, on järkevää ja kustannustehokasta laajentaa testausautomaatio integraatiotestaukseen jatkamalla tämän automatisointikehyksen käyttöä.

Käyttöliittymiin perustuvien testiskriptien luomisen ja ylläpidon kustannuksia pienentämään on kehitetty koneoppimista ja tekoälyä hyödyntäviä automatisointityökaluja. Koneoppimista hyödyntävien testiskriptien on tietyissä rajoissa mahdollista ylläpitää itse itsensä käyttöliittymäkoodia koskevien muutosten jälkeen, sillä ne käyttävät elementtien tunnistamisessa kaikkia saatavilla olevia attribuutteja eikä ainoastaan yksilöllisiä tunnisteita tai muutaman attribuutin yhdistelmiä. Tällaisia ratkaisuja on viime vuosina tullut markkinoille useita, esim. Testim, TestCraft, Mabl (Sam, 2020). Laskentatehovaatimustensa vuoksi nykyiset ratkaisut hyödyntävät pilvialustaa. Koneoppimista ja tekoälyä hyödyntävien automatisointityökalujen käyttö on integraatiotestauksen automatisoinnin yksi varteenotettava vaihtoehto erityisesti silloin, kun työkalut voi asentaa paikallisesti (On-Premise) eikä pilvialustan käyttö ole pakollista.

Edustaja- ja toimielintietojen hallintajärjestelmä on eduskunnan vanhin käytössä oleva tietojärjestelmä, eikä sillä ole mm. selainpohjaista käyttöliittymää. Sen toimintojen automatisointi selainpohjaisille käyttöliittymille suunnitelluilla automatisointityökaluilla ei täten ole mahdollista. Edustaja- ja toimielintietojen tilastojen julkaisun automatisointi voidaan toteuttaa integraatiojärjestelmän tarjoaman BTM (Business Transaction Monitoring) työkalun avulla, koska testiaineistona voidaan käyttää edellisenä yönä syntyneitä tilastoja. Testiaineistoa ei näin ollen tarvitse luoda järjestelmän käyttöliittymän kautta vaan integraatiojärjestelmä voi lähettää yöllä vastaanottamansa tilastotiedot uudelleen julkaisujen hallintajärjestelmälle testausta varten luodun sanomavirran (Message flow) avulla. Tilastotietojen julkaisussa käytetään samaa edustaja- ja toimielintietojen hallintajärjestelmän ja julkaisujen hallintajärjestelmän välistä integraatiota, jota käytetään testaajan käyttöliittymässä käynnistämissä yksittäisissä edustaja- ja toimielintietojen julkaisuissa ja joka on yksi kriittisistä integraatioista. Automatisoimalla tilastotietojen julkaisusanomien uudelleenkäsittely integraatiojärjestelmässä saadaan kriittisen integraation testaukseen kuluvaa aikaa vähennettyä merkittävästi. Näin manuaalisesti testattavaksi jää ainoastaan edustaja- ja toimielintietojen hallintajärjestelmän ja integraatiojärjestelmän välinen yhteys, jonka lopputulos voidaan todentaa nopeammin lokilta kuin tarkastelemalla lopputulosta julkaisujen hallintajärjestelmästä. Koska edustaja- ja toimielintietojen hallintajärjestelmä on uusittava lähitulevaisuudessa, ei järjestelmän integraatioiden testausta ole järkevää automatisoida sen vanhan käyttöliittymäsovelluksen toimintoja automatisoimalla.

Integraatiojärjestelmän tarjoamien työkalujen avulla ei voida automatisoida eduskunnan tietojärjestelmien integraatiotestausta, sillä työkalujen käyttö rajoittuu vain integraatiojärjestelmän

toimintaan eikä testaa integraatioita päästä päähän. Eduskunnan tietojärjestelmien vaatimusten mukainen toiminta integraatioiden kautta välitettävien palvelujen tuottamisessa ja hyödyntämisessä sekä eduskunnan järjestelmien järjestelmän mission toteutuminen jäävät varmistamatta, mikäli integraatioita ei testata päästä päähän. BTM-työkalun käyttö on kuitenkin tärkeää integraatiojärjestelmän järjestelmätestauksen automatisoinnissa, sillä tätä ei voida automatisoida käyttöliittymiin perustuvalla automaatiolla.

Palveluiden virtualisointi ja jäljitelmät eivät ole yksinään riittäviä vaihtoehtoja eduskunnan integraatiotestauksen automatisointiin, sillä ne eivät testaa todellisia integraatioita eivätkä verifioi ydinjärjestelmien yhteistoimintaa. Ne eivät varmista eduskunnan tietojärjestelmien vaatimusten mukaista toimintaa integraatioiden kautta välitettävien palvelujen osalta eikä mission toteutumista. Jäljitelmät auttavat integraatiotestauksessa kuitenkin silloin, jos jokin ydinjärjestelmä ei ole saatavissa. Jäljitelmiä integraatiotestausta varten voidaan toteuttaa esim. SoapUI työkalulla, jota on hyödynnetty täysistunnon hallintajärjestelmän ja valtiopäiväasioiden ja –asiakirjojen hallintajärjestelmän välisen integraation korvaamisessa. Jäljitelmät ovat riittäviä ja tarpeellisia yksittäisen järjestelmän järjestelmätestauksen automatisoinnissa, mutta ne eivät korvaa integraatioiden todellista testausta.

Eduskunnan jatkuvan integraation Jenkins-palvelinta voidaan käyttää automatisoitujen integraatiotestien ajamiseen järjestelmien kehitystyössä. Jäljitelmien avulla järjestelmien omat testit on mahdollista ajaa joka koodimuutoksen yhteydessä, mutta automatisoidut integraatiotestit voidaan ajaa ainoastaan silloin kun järjestelmät ovat testiympäristössä saatavissa. Automatisoidut integraatiotestit voidaan ajaa Jenkins-palvelimella ajastetusti ja palvelimelta saadaan nopeasti palaute testauksesta. Jenkins-palvelimelle on asennettavissa oma Robot Framework lisäosa, jonka avulla testiskriptit on ajettavissa ja myös tulokset saadaan tällöin näkyviin Jenkinsin kautta. Käyttöliittymiin perustuvien integraatiotestiskriptien suoritus on mahdollista tehdä myös erillisellä tätä varten asennetulla palvelimella tai tietokoneella, johon on asennettu suoritukseen tarvittavat ohjelmat ja jolta on pääsy testiympäristöön järjestelmien käyttöliittymiin.

JIRA-järjestelmän käyttöä integraatiotestauksen suunnittelussa, suorituksessa ja raportoinnissa voidaan tehostaa erilaisin lisäosin. Esimerkiksi XRAY, Zephyr ja TestManagement ovat testitapausten ja testauksen hallinnan parantamiseen tarkoitettuja lisäosia (Atlassian, 2020).

7.3 Ratkaisujen evaluointi

Integraatiotestauksen automatisointia kokeiltiin ensin tilastotietojen julkaisun automatisoinnilla, sillä tämä ei vaatinut toimintojen testaamista palvelua tuottavassa tai käyttävässä järjestelmässä vaan ainoastaan integraatiojärjestelmässä. BTM-työkalun avulla toteutetussa testiautomaatiossa edustaja- ja toimielintietojen hallintajärjestelmän sekä valtiopäiväasioiden ja -asia-

kirjojen hallintajärjestelmän edellisenä yönä lähettämät tilastojen julkaisusanomat otettiin integraatiojärjestelmän käsiteltyjen sanomien jonosta uudelleen käsittelyyn ja lähetettiin julkaisujen hallintajärjestelmään. Testauksen lopputulos tarkistettiin julkaisujen hallintajärjestelmän vastaanottokuittauksen tarkistuksella integraatiojärjestelmän lokilta.

Testiautomaatio BTM-työkalun avulla osoittautui toimivaksi tilastotietojen julkaisun testaukseen, mutta julkaisusanomien tunnistaminen käsiteltyjen sanomien jonosta osoittautui haasteelliseksi. Järjestelmät lähettävät myös muita sanomia yöaikaan, joten sanomien käsittelyaikaa ei voinut hyödyntää sanomien tunnistuksessa. Sanomakehyksien yksilölliset tunnisteet puolestaan sisälsivät merkkejä, joita BTM ei voinut lukea. Ratkaisu vaati muutoksia tilastotiedot lähettäviin järjestelmiin, jotta ne muodostavat tilastotietojen julkaisusanomiin yksilöivät tunnisteet BTM-työkalun hyväksymässä muodossa. BTM-työkalulla toteutettua testausautomaatiota ei voida tilastotietojen julkaisun lisäksi laajentaa eduskunnan järjestelmien integraatiotestaukseen, mutta sen käytettävyyttä integraatiojärjestelmän oman järjestelmätestauksen automatisointiin tulee vielä arvioida.

Integraatiotestauksen automatisointia kokeiltiin laajentamalla täysistunnon hallintajärjestelmässä ja valtiopäiväasioiden ja –asiakirjojen hallintajärjestelmässä toteutettua testausautomaatiota avainskenaarioiden mukaisten käyttötapausten testaukseen. Kokeilu aloitettiin yhteisillä työpajoilla valtiopäiväasioiden ja -asiakirjojen hallintajärjestelmän ja täysistunnon hallintajärjestelmän aikaisemmin toteutettua testausautomaatiota toteuttamassa olleiden ylläpitäjien kanssa. Työpajoihin osallistui myös järjestelmien käyttöliittymien koodin hyvin tuntevat järjestelmäkehittäjät. Kokeilussa automatisoitiin Robot Framework automatisointikehyksellä luvussa 6.1 ja taulukossa 3 kuvatut avainskenaarioiden käyttötapaukset, jotka testaavat kaikki muut kuvassa 15 esitetyt kriittiset korkeimman prioriteetin integraatiotestitapaukset paitsi edustaja- ja toimielintietojen hallintajärjestelmän integraatiota julkaisujen hallintajärjestelmään. Kokeilun ulkopuolelle jätettiin testausautomaation lopputulosten automaattinen todentaminen järjestelmien käyttöliittymistä käsin. Kokeilun tuloksena kaikki eduskunnan kriittiset integraatiotestit saatiin automatisoitua 20 henkilötyöpäivässä. Kriittiset integraatiotestit voitiin kokeilun jälkeen suorittaa noin 20 minuutissa ja testauksen suunnitteluun ja organisointiin ei tarvinnut käyttää aikaa. Kriittisten integraatiotestien suoritukseen tarvittiin ainoastaan yksi testaaja, joka käynnisti testiskriptien ajon ja raportoi testiskriptien tulokset. Aiemmin kriittisten integraatioiden testaus vaati viiden testaajan työn ja noin neljä tuntia aikaa.

Integraatiotestauksen automatisoinnissa voitiin käyttää uudelleen aiemmin luotuja testiskriptejä Robot Framework automatisointikehyksen modulaarisuuden ja skriptien uudelleenkäytettävyyden vuoksi. Täysistunnon hallintajärjestelmän käyttöliittymän koodiin jouduttiin tekemään pieniä muutoksia elementtien yksilöivän nimeämiskäytännön toteuttamiseksi. Automa-

tisoituja integraatiotestejä ajettiin useamman kerran eri ajanjaksoina tapahtuneissa integraatiotestauksissa keväällä ja kesällä 2019. Tuona aikana ylläpitotarpeita testiskripteihin ei tarvittu, sillä muutokset järjestelmissä eivät kohdistuneet eivätkä vaikuttaneet järjestelmien käyttöliittymäkoodiin.

Integraatiotestauksessa jouduttiin kriittisten integraatioiden testauksen osalta suorittamaan manuaalisesti ainoastaan testausten lopputulosten tarkistaminen käyttöliittymistä, mikä vei aikaa noin 10 minuuttia. Loput integraatiotestit suoritettiin manuaalisesti kuten ennenkin. Kaikkien integraatioiden testaus kesti alle kaksi tuntia ja vaati testiskriptien käynnistäjän lisäksi ainoastaan kahden testaajan työn, edustaja- ja toimielintietojen hallintajärjestelmän testaajan ja valtiopäiväasioiden ja -asiakirjojen hallintajärjestelmän testaajan. Kokeilussa ei hyödynnetty Jenkins-palvelinta, vaan integraatiotestiskriptit ajettiin kokeilussa erilliseltä tähän tarkoitukseen asennetulta tietokoneelta, jonka määritettyyn hakemistoon testauksen raportit syntyivät. Integraatiotestiskriptin suorituksen aikana eduskunnan testaajat pystyivät omissa työpisteissään seuraamaan omilta koneiltaan järjestelmien käyttöliittymien kautta testauksen suorituksen kulkua ja lopulta testauksen lopputuloksia. Kokeilun perusteella Robot Framework soveltuu eduskunnan integraatiotestauksen automatisoinnin ratkaisuksi.

Ohjelmistorobotiikka on IEEE:n standardin mukaan ennakkoon konfiguroitu ohjelmiston esiintymä, joka liiketoiminnan sääntöjä ja ennalta määritettyjä toimintoja käyttämällä täydentää yhden tai useamman järjestelmän yhdistettyjen prosessien, aktiviteettien, transaktioiden ja tehtävien suorituksen tuottaakseen ihmisen suorittaman poikkeushallinnan sisältävän palvelun (IEEE Std 2755-2017, 2017, s. 11). Ohjelmistorobotiikka pyrkii automatisoimaan työnkulkuja jäljittelemällä käyttäjän toimia käyttölittymässä halutun lopputuloksen saavuttamiseksi (S. Yatskiv, Voytyuk, N. Yatskiv, Kushnir, Trufanova & Panasyuk, 2019, s. 293). Testauksen automatisointiin ja ohjelmistorobotiikkaan soveltuvan Robot Framework automatisointikehyksen avulla toteutetut eduskunnan integraatiotestit todentavat vaatimusten mukaisen vuorovaikutuksen järjestelmien välillä, sillä robotti testaa avainskenaarioihin sisältyvät integraatiot käyttöliittymien kautta täsmälleen kuten eduskunnan tietojärjestelmien pääkäyttäjät ne testaavat. Robotin suorittama testaus ei ole altis inhimillisistä syistä johtuviin vaihteluihin testauksen suorituksessa. Robotti tuottaa heti testauksen suorituksen jälkeen raportin, josta testaustulokset ovat nopeasti tarkistettavissa. Ohjelmistorobotiikan työkalut ovat monikäyttöisiä ja niitä voi käyttää niin käyttöliittymään perustuvissa testeissä kuin testiaineiston valmistelussa, testauksen esiehtojen asettamisessa, regressiotestauksessa ja odotettujen lopputulosten tarkistamisessa (S. Yatskiv, Voytyuk, N. Yatskiv, Kushnir, Trufanova & Panasyuk, 2019, s. 294).

Käyttöliittymiin perustuvan testausautomaation merkittävimpänä haittapuolena nähdään testien ylläpitotarve ja ylläpidon kustannukset. Ylläpitotarve käyttöliittymiin perustuvissa järjes-

telmien järjestelmän integraatiotesteissä nousee yksittäisten järjestelmien käyttöliittymiin kohdistuneista muutoksista. Robot Framework automatisointikehyksen modulaarisuuden ja skriptien uudelleenkäytettävyyden vuoksi järjestelmien integraatiotestien ylläpidossa voidaan jatkokäyttää yksittäisten järjestelmien Robot Framework automatisointikehyksellä ylläpidettyjä testien osia. Integraatiotestejä on mahdollista ylläpitää osissa, jolloin tiettyyn yksittäiseen järjestelmään kohdistuvien testien osia ylläpitävät kyseisen järjestelmän testien ylläpitäjät. Tällöin ylläpito on myös tehokkainta, sillä yksittäisen järjestelmän testien ylläpitäjillä on paras osaaminen testien osien ylläpitoon sekä paras tietämys käyttöliittymiin kohdistuneista testien ylläpitoa vaativista muutoksista. Integraatiotestien ylläpitäjän tehtäväksi jää järjestelmien ylläpitäjien ylläpitämien testien osien jatkokäyttö ja yhdistäminen integraatiotesteiksi. Yhteistyö järjestelmien testien ylläpitäjien välillä on tärkeää, ja integraatiotestien ylläpidon tehtävien sekä vastuiden tulee olla selkeät ja sovitut. Jenkins-palvelimelle ajastettujen integraatiotestien avulla ylläpitotarpeet huomataan nopeasti ja ylläpitotehtävät tiedetään käynnistää sovitun ylläpitoprosessin mukaisesti.

8 Johtopäätökset

Tässä tutkielmassa on tarkasteltu tieteellisen tutkimuksen ja ammattikirjallisuuden pohjalta keinoja ja näkökohtia järjestelmien järjestelmän integraatiotestauksen automatisointiin sekä arvioitu näiden soveltuvuutta eduskunnan lainsäädäntötyön tietojärjestelmien välisten integraatioiden testauksen automatisointiin. Tutkielman tavoitteena oli löytää eduskunnan tietojärjestelmien integraatiotestauksen automatisointiin sopivat vaihtoehdot, joiden pohjalta integraatiotestauksen automatisointi voidaan toteuttaa.

Järjestelmien järjestelmässä integraatiotestauksen automatisoinnissa huomioitavia näkökohtia ovat erillisten järjestelmien toiminnallinen ja hallinnollinen itsenäisyys ja tästä aiheutuvat haasteet. Itsenäisten järjestelmien ohjelmakoodi ei ole järjestelmien järjestelmässä saatavissa, minkä vuoksi integraatiotestauksen automatisointi voi perustua ainoastaan itsenäisten järjestelmien käyttöliittymiin ja käyttöliittymien kautta saavutettavaan koodiin. Eduskunnan tietojärjestelmien integraatiotestauksen automatisoinnin vaihtoehtojen evaluointia varten integraatiotestauksen automatisointia kokeiltiin testauksen automatisointiin ja ohjelmistorobotiikkaan soveltuvan modulaarisen Robot Framework automatisointikehyksen avulla. Kokeilusta saatujen tulosten perusteella testauksen automatisointiin ja ohjelmistorobotiikkaan soveltuvan automatisointikehyksen avulla on mahdollista automatisoida eduskunnan tietojärjestelmien integraatiotestaus. Ohjelmistorobotiikan avulla voidaan jäljitellä eduskunnan testaajien järjestelmien käyttöliittymien kautta suorittamaa integraatiotestausta ja näin varmistaa vaatimusten mukaisen vuorovaikutuksen toteutuminen järjestelmien välillä. Robot Framework automatisointikehyksen avulla toteutetun testausautomaation ja ohjelmistorobotiikan avulla integraatiotestit saadaan suoritettua manuaalista testausta merkittävästi nopeammin ja vähemmillä resursseilla.

Kirjallisuudessa esitettyjä järjestelmien integraatiotestauksen automatisoinnin muita keinoja, jotka soveltuvat eduskunnan tietojärjestelmien integraatiotestauksen automatisointiin ovat testauksen hallinnan työkalujen tehokkaampi käyttö sekä jatkuvan integraation käyttäminen. Eduskunnan tietojärjestelmien integraatiotestauksen hallinnan työkalun käyttöä on mahdollista tehostaa testauksen suunnittelua ja raportointia parantavalla lisäosalla. Automatisoidut integraatiotestit voidaan ajaa eduskunnan jatkuvan integraation Jenkins-palvelimella.

Lähteet

- Alégroth, E., Feldt, R., & Kolström, P. (2016). Maintenance of automated test suites in industry: An empirical study on Visual GUI Testing. Information and Software Technology, Volume 73, 2016.
- Alégroth, E., Karlsson, A., & Radway, A. (2018). Continuous Integration and Visual GUI Testing: Benefits and Drawbacks in Industrial Practice. Software Testing Verification and Validation (ICST) 2018 IEEE 11th International Conference on, pp. 172-181, 2018.
- Ali, N., B., Petersen, K., & Mäntylä, M., V. (2012). Testing highly complex system of systems: An industrial case study, Proceedings of the 2012 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement, Lund, 2012, pp. 211-220.
- Atlassian. (2020, February 17). *Using JIRA Software for Test Case Management*. https://confluence.atlassian.com/jirakb/using-jira-software-for-test-case-management-136872198.html
- Berner, S., Weber, R., & Keller, R. K. (2005, May). Observations and lessons learned from automated testing. In Software Engineering, 2005. ICSE 2005. Proceedings. 27th International Conference on (pp. 571-579). IEEE.
- Clark, J., O. (2009). System of Systems Engineering and Family of Systems Engineering from a standards, V-Model, and Dual-V Model perspective, 2009 3rd Annual IEEE Systems Conference, Vancouver, BC, 2009, pp. 381-387
- Collins, E., F., & de Lucena, V., F. (2012). Software Test Automation practices in agile development environment: An industry experience report, 2012 7th International Workshop on Automation of Software Test (AST), Zurich, 2012, pp. 57-63
- Dahmann, J., Lane, J., A., Rebovich, G., & Lowry, R. (2010). Systems of systems test and evaluation challenges, 2010 5th International Conference on System of Systems Engineering, Loughborough, 2010, pp. 1-6.
- Dustin, E., Rashka, J., & Paul, J. (1999). Automated software testing: introduction, management, and performance. Addison-Wesley Professional.

- Eniser, H., F., Sen, A., & Polat, S., O. (2018). Fancymock: creating virtual services from transactions. In Proceedings of the 33rd Annual ACM Symposium on Applied Computing (SAC ’18). Association for Computing Machinery, New York, USA
- Gandhi, P. (2016, March). A survey on prospects of automated software test case generation methods. In Computing for Sustainable Global Development (IN-DIACom), 2016 3rd International Conference on (pp. 3867-3871). IEEE.
- Hu, G., Zhu, L. & Yang, J. (2018). AppFlow: Using Machine Learnin to Synthesize Robust, Reusable UI Tests, in Proceedings of the 26th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE ’18), November 4-9, 2018, Lake Buena Vista, FL, USA, pp. 269-282.
- IEEE Std. 2755-2017: IEEE Guide for Terms and Concepts in Intelligent Process Automation, United States of America: IEEE Standards Association, 2017
- International Software Testing Qualifications Board [ISTQB]. (2017, March 17). *ISTQB:n testaussanasto v. 2.3 Suomi – Englanti*. http://www.fistb.fi/sites/fistb/files/liitteet/istqb_sanasto_2015-04-30%202.3%20FI-ENG.pdf
- ISO/IEC, ISO/IEC 25010:2010 SOFTWARE ENGINEERING - SOFTWARE PRODUCT QUALITY REQUIREMENTS AND EVALUATION (SQUARE) – SYSTEM AND SOFTWARE QUALITY MODELS, ISO/IEC JTC 1/SC 7, 2010.
- ISO/IEC/IEEE International Standard - Software and systems engineering -- Software testing -- Part 5: Keyword-Driven Testing, in ISO/IEC/IEEE 29119-5 First edition 2016-11-15 , pp.1-69, 2016
- Karhu, K., Repo, T., Taipale, O., & Smolander, K. (2009, April). Empirical observations on software testing automation. In Software Testing Verification and Validation, 2009. ICST'09. International Conference on (pp. 201-209). IEEE.

- Kasurinen, J., Taipale, O., & Smolander, K. (2010). Software test automation in practice: empirical observations. Advances in Software Engineering, 2010.
- Kazman, R., Nielsen, C., & Schmid, K. (2013). Understanding Patterns for System-of-Systems Integration (CMU/SEI-2013-TR-017). Retrieved September 21, 2018, from the Software Engineering Institute, Carnegie Mellon University website: http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID=75750
- Lewis, G., Morris, E., Place, P., Simanta, S., Smith, D., & Wrage, L. (2008). Engineering Systems of Systems, 2008 2nd Annual IEEE Systems Conference, Montreal, Que., 2008, pp. 1-6.
- Lima, B. (2018). Automated scenario-based integration testing of distributed systems. In Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE 2018). Association for Computing Machinery, New York, USA, Pages 956–958.
- Linz, T. (2014). *Testing in Scrum: A Guide for Software Quality Assurance in the Agile World* . Rocky Nook Inc.
- Müller, T., Beer, A., Klonk, M., & Verma, R. (2017, March 16). *Foundation Level Syllabus, International Software Testing Qualifications Board, Versio 2010*. http://www.fistb.fi/sites/fistb/files/liitteet/FL%20Syllabus%2020101123_0.pdf
- Neves, V., Angelis, G., Bertolino, A. & Garcés, L.(2018). Do we need new strategies for testing Systems-of-Systems?, 2018 ACM/IEEE 6th International Workshop on Software Engineering for Systems-of-Systems
- Nurmeksela, R. (2006). Suomalaisen lainsäädäntötyön tiedonhallinta: Suuntana semanttinen web. Helsinki: Eduskunnan kanslia.
- Pinto, L., S., Sinha, S., & Orso, A. (2012). Understanding myths and realities of test-suite evolution. In Proceedings of the ACM SIGSOFT 20th International Symposium on the Foundations of Software Engineering. ACM, 33.
- Robot Framework. (2020, February 16). *SeleniumLibrary* https://robotframework.org/SeleniumLibrary/SeleniumLibrary.html

- Robot Framework. (2020, February 9). *Robot Framework Introduction.* https://robotframework.org/#introduction
- Sam. (2020, February 16). *10 Most Popular AI Powered Test Automation Tools.* Medium. https://medium.com/@softwaretest/10-most-popular-ai-powered-test-automation-tools-597e43436c9a
- Shenoy, S. T., Mohapatra, A., & Swamy, R. (2014). A multi-dimensional testing approach for an ESB. International Journal of Software Engineering and Technology, 1(2).
- Silva, E., Batista, T., & Oquendo, F. (2015) A mission-oriented approach for designing system-of-systems, 2015 10th System of Systems Engineering Conference (SoSE), San Antonio, TX, 2015, pp. 346-351.
- SoapUI (2020, February 24). *Getting Started with SoapUI Mocking.* https://www.soapui.org/getting-started/mocking.html
- Software Testing Help. (2020, February 23). *List Of The Best Test Management Tools In 2020*. https://www.softwaretestinghelp.com/15-best-test-management-tools-for-software-testers/
- Software Testing Help. (2020, February 24). *7 Best Service Virtualization Tools in*. https://www.softwaretestinghelp.com/service-virtualization-tools/
- Spillner, A., Linz, T., & Schaefer, H. (2014). Software testing foundations: a study guide for the certified tester exam. Rocky Nook, Inc.
- Suomen perustuslaki 11.6.1999. 731/1999. https://www.finlex.fi/fi/laki/alkup/1999/19990731
- Thummalapenta, S., Sinha, S., Singhania, N., & Chandra, S. (2012, June). Automating test automation. In Software Engineering (ICSE), 2012 34th International Conference on (pp. 881-891). IEEE.
- Yatskiv, S., Voytyuk, I., Yatskiv, N., Kushnir, O., Trufanova, Y., & Panasyuk, V. (2019). Improved Method of Software Automation Testing Based on the Robotic Process Automation Technology. In 2019 9th International Conference on Advanced Computer Information Technologies (ACIT) (pp. 293-296). IEEE.
- Zapata, F., Akundi, A., Pineda, R., & Smith, E. (2013). Basis Path Analysis for

Testing Complex System of Systems. Procedia Computer Science, Volume 20, 2013, Pages 256-261.

Liite 1. Eduskunnan ydinjärjestelmien integraatiot ja tietovirrat (Luottamuksellinen)